Constrained Keys for Invertible Pseudorandom Functions

Dan Boneh, Sam Kim(✉), and David J. Wu

Stanford University, Stanford, USA
skim13@cs.stanford.edu

Abstract. A constrained pseudorandom function (PRF) is a secure PRF for which one can generate constrained keys that can only be used to evaluate the PRF on a subset of the domain. Constrained PRFs are used widely, most notably in applications of indistinguishability obfuscation ($i\mathcal{O}$). In this paper we show how to constrain an *invertible* PRF (IPF), which is significantly harder. An IPF is a secure *injective* PRF accompanied by an inversion algorithm. A constrained key for an IPF can only be used to evaluate the IPF on a subset S of the domain, and to invert the IPF on the image of S. We first define the notion of a constrained IPF and then give two main constructions: one for puncturing an IPF and the other for (single-key) circuit constraints. Both constructions rely on recent work on *private* constrained PRFs. We also show that constrained pseudorandom permutations for many classes of constraints are impossible under our definition.

1 Introduction

Pseudorandom functions (PRFs) [34] and pseudorandom permutations (PRPs) [41] have found numerous applications in cryptography, such as encryption, data integrity, user authentication, key derivation, and others. Invertible PRFs are a natural extension that borrows features from both concepts. An invertible PRF (IPF) is an efficiently-computable *injective* function $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ equipped with an efficient inversion algorithm $\mathsf{F}^{-1} : \mathcal{K} \times \mathcal{Y} \to \mathcal{X} \cup \{\perp\}$. The inversion algorithm is required to satisfy the following two properties for all $k \in \mathcal{K}$:

- (1) $\mathsf{F}^{-1}\big(k,\ \mathsf{F}(k,x)\big) = x$ for all $x \in \mathcal{X}$.
- (2) $\mathsf{F}^{-1}(k,y) = \perp$ whenever y is not in the image of $f(x) := \mathsf{F}(k,x)$.

We say that an IPF F is secure if no poly-bounded adversary can distinguish the following two experiments. In one experiment the adversary is given oracles for the function $f(x) := \mathsf{F}(k,x)$ and its inverse $f^{-1}(x) := \mathsf{F}^{-1}(k,x)$, where k is randomly chosen in $\mathcal{K}$. In the other experiment, the adversary is given oracles for a *random* injective function $g : \mathcal{X} \to \mathcal{Y}$ and its inverse $g^{-1} : \mathcal{Y} \to \mathcal{X} \cup \{\perp\}$. These two experiments should be indistinguishable. We define this in detail in

The full version of this paper is available at https://eprint.iacr.org/2017/477.pdf.

Y. Kalai and L. Reyzin (Eds.): TCC 2017, Part I, LNCS 10677, pp. 237–263, 2017.
https://doi.org/10.1007/978-3-319-70500-2_9

Sect. 3. Note that when $\mathcal{X} = \mathcal{Y}$, an IPF is the same as a strong pseudorandom permutation [41].

IPFs come up naturally in the context of deterministic authenticated encryption (DAE) [50], as discussed below. A closely related concept called a *pseudorandom injection* (PRI) [50] is similar to an IPF except for some syntactic differences (an IPF is a pseudorandom injection without additional length constraints and with an empty header).

Constrained PRFs. In this paper we define and construct constrained IPFs. It is helpful to first review constrained PRFs [19,21,40]. Recall that a PRF $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is said to be a constrained PRF if one can derive constrained keys from the master PRF key k. A constrained key k_g is associated with a predicate $g\colon \mathcal{X} \to \{0,1\}$, and this k_g enables one to evaluate $F(k,x)$ for all $x \in \mathcal{X}$ where $g(x) = 1$, but at no other points of $\mathcal{X}$. A constrained PRF is secure if given constrained keys for predicates $g_1, \ldots, g_Q$ of the adversary's choosing, the adversary cannot distinguish the PRF from a random function at points not covered by the given keys, namely at points x where $g_1(x) = \cdots = g_Q(x) = 0$. We review the precise definition in Sect. 3.1.

Constrained PRFs have found numerous applications in cryptography [19,21,40]: they imply identity-based key exchange and broadcast encryption, and are a crucial ingredient in many applications of indistinguishability obfuscation ($i\mathcal{O}$) [51].

The simplest non-trivial constraint is a *puncturing* constraint, a constraint that enables one to evaluate the function on its entire domain except for one point. For $x \in \mathcal{X}$ we denote by k_x a *punctured key* that lets one evaluate the PRF at all points in $\mathcal{X}$, except for the punctured point x. Given the key k_x, the adversary should be unable to distinguish $F(k,x)$ from a random element in $\mathcal{Y}$. PRFs supporting puncturing constraints can be easily constructed from the tree-based PRF of [34], as discussed in [19,21,40].

Constrained IPFs. Given the wide applicability of constrained PRFs, it is natural to look at constraining other symmetric primitives such as PRPs and, more generally, IPFs. A constrained key k_g for an IPF enables one to evaluate the IPF at all points $x \in \mathcal{X}$ for which $g(x) = 1$, and invert at all points $y = \mathsf{F}(k,x') \in \mathcal{Y}$ for which $g(x') = 1$. Security for a constrained IPF is defined as for a PRF: the adversary is given a number of constrained keys and tries to distinguish the IPF from a random injective function at points not covered by any of the given keys. See Sect. 3.1 for more details.

We first show in Sect. 3.3 that constrained PRPs for many constraint classes do not exist in our model. However constrained IPFs, where the range can be larger than the domain, can exist. The challenge is to construct them. Surprisingly, constraining an IPF is significantly harder than constraining a PRF, even for simple puncturing constraints. For example, it is not difficult to see that puncturing a Luby-Rackoff cipher by puncturing the underlying PRFs does not work.

In this paper, we present constrained IPFs for both puncturing constraints and for arbitrary circuit constraints. Both constructions make use of a recent primitive called a *private constrained* PRF [18] that can be constructed from the learning with errors (LWE) problem [15,22,24]. Roughly speaking, a private constrained PRF is a constrained PRF where a constrained key k_g reveals nothing about the constraint g. Before we describe our constructions, let us first look at an application.

IPFs and deterministic encryption. While constrained IPFs are interesting in their own right, they come up naturally in the context of deterministic encryption. IPFs are related to the concept of *deterministic authenticated encryption* (DAE) introduced by Rogaway and Shrimpton [50] where encryption is deterministic and does not take a nonce as input. A DAE provides the same security guarantees as (randomized) authenticated encryption, as long as all the messages encrypted under a single key are distinct. Rogaway and Shrimpton show that an IPF whose range is sufficiently larger than its domain is equivalent to a secure DAE. They further require that the length of the IPF output depend only on the length of the input, and this holds for all our constructions. Hence, our constrained IPFs give the ability to constrain keys in a DAE encryption scheme: the constrained key holder can only encrypt/decrypt messages that satisfy a certain predicate.

1.1 Building Constrained IPFs

In Sect. 4, we present two constructions for constrained IPFs on a domain $\mathcal{X} = \{0,1\}^n$. Our first construction, a warm-up, only supports puncturing constraints. Our second construction gives a constrained IPF for arbitrary circuit constraints, but is only secure if a single constrained key is released. Here we give the main ideas behind the constructions. Both rely heavily on the recent development of private constrained PRFs. In Sect. 5, we show how to instantiate our constructions from the LWE assumption. In Sect. 7, we also show that using $i\mathcal{O}$, it is possible to construct a multi-key, circuit-constrained IPF.

A puncturable IPF. Let $\mathsf{F}_1 \colon \mathcal{K}_1 \times \mathcal{X} \to \mathcal{V}$ and $\mathsf{F}_2 \colon \mathcal{K}_2 \times \mathcal{V} \to \mathcal{X}$ be two secure PRFs. Define the following IPF F on domain $\mathcal{X}$ using a key $k = (k^{(1)}, k^{(2)}) \in \mathcal{K}_1 \times \mathcal{K}_2$:

$$\mathsf{F}\big((k^{(1)}, k^{(2)}),\ x\big) := \left\{\begin{array}{l} y_1 \leftarrow \mathsf{F}_1(k^{(1)}, x) \\ y_2 \leftarrow x \oplus \mathsf{F}_2(k^{(2)}, y_1) \\ \text{output } (y_1, y_2) \end{array}\right\} \qquad \mathsf{F}^{-1}\big((k^{(1)}, k^{(2)}),\ (y_1, y_2)\big) := \left\{\begin{array}{l} x \leftarrow \mathsf{F}_2(k^{(2)}, y_1) \oplus y_2 \\ \text{if } \mathsf{F}_1(k^{(1)}, x) \neq y_1 \\ \quad \text{then } x \leftarrow \perp \\ \text{output } x \end{array}\right\} \tag{1.1}$$

It is not difficult to show that F is a secure IPF. In fact, one can view this IPF as an instance of a DAE construction called SIV (Synthetic-IV) [50].

The question is how to securely puncture F. As a first attempt, suppose F_1 is a puncturable PRF, say constructed from the tree-based GGM construction [34].

To puncture the IPF F at a point $x \in \mathcal{X}$, one can puncture F_1 at x to obtain the IPF punctured key $k_x := (k_x^{(1)}, k^{(2)})$. This key k_x prevents the evaluation F at the point x, as required. However, this is completely insecure. To see why, observe that given k_x, the adversary can easily distinguish $\mathsf{F}(k, x)$ from a random pair in $\mathcal{V} \times \mathcal{X}$: given a challenge value (y_1, y_2) for $\mathsf{F}(k, x)$, the adversary can simply test if $x = \mathsf{F}_2(k^{(2)}, y_1) \oplus y_2$. This will be satisfied by $\mathsf{F}(k, x)$, but is unlikely to be satisfied by a random pair in $\mathcal{V} \times \mathcal{X}$.

To properly puncture F at x we must puncture F_1 at x *and* puncture F_2 at $y_1 := \mathsf{F}_1(k^{(1)}, x)$. The punctured key for F is then $k_x := (k_x^{(1)}, k_{y_1}^{(2)})$. Here, it is vital that the punctured key $k_{y_1}^{(2)}$ reveal nothing about the punctured point y_1. Otherwise, it is again easy to distinguish $\mathsf{F}(k, x) = (y_1, y_2)$ from a random pair in $\mathcal{V} \times \mathcal{X}$ using the exposed information about y_1. To ensure that y_1 is hidden, we must use a *private puncturable* PRF for F_2. Currently the best constructions for a private puncturable PRF rely on the LWE assumption [15,22,24]. It is not known how to construct a private puncturable PRF from one-way functions. We show in Theorem 4.3 that with this setup, the puncturable IPF in (1.1) is secure.

A constrained IPF for circuit constraints. Next we generalize (1.1) to support an arbitrary circuit constraint g. As a first step we can constrain $k^{(1)}$ to g so that the IPF constrained key is $k_g := (k_g^{(1)}, k^{(2)})$. We can use for F_1 any of the candidate circuit-constrained PRFs [19,23].

As before, this is insecure: for security we must also constrain F_2. However we immediately run into a problem. Following the blueprint in (1.1) we must puncture F_2 at all points $\mathsf{F}_1(k^{(1)}, x)$ where $g(x) = 0$. However, because the size of this set can be super-polynomial, we would need to constrain F_2 to a set containing super-polynomially-many pseudorandom points. The difficulty is that F_2 cannot efficiently test if an input $v \in \mathcal{V}$ satisfies $v = \mathsf{F}_1(k^{(1)}, x)$ with $g(x) = 0$. Because F_1 is not invertible, this cannot be done even given $k^{(1)}$.

We solve this problem by replacing $\mathsf{F}_1(k^{(1)}, x)$ with a CCA-secure public-key encryption $\mathsf{PKE.Encrypt}(\mathsf{pk}, x; r_x)$, where the randomness $r_x = \mathsf{F}_1(k^{(1)}, x)$ is derived from F_1 and pk is the public key. In this case, the input to F_2 is a ciphertext ct that encrypts the point x. The output of the IPF is the pair $(\mathsf{ct},\ \mathsf{F}_2(k^{(2)}, \mathsf{ct}) \oplus x)$. When constraining F_2, we embed the secret decryption key sk for the public-key encryption scheme in the constrained key. Then, on an input ciphertext ct, the constraint function first decrypts ct (using sk) to obtain a value $x \in \mathcal{X}$, and then checks if $g(x) = 1$. Because knowledge of sk allows one to invert on *all* points, it is *critical* that the constrained key hides sk. Here, we rely on a strong simulation-based notion of constraint privacy [15,24]. In Theorem 4.7, we show that as long as the underlying PKE scheme is CCA-secure and F_2 is a (single-key) *private* constrained PRF, then the resulting scheme is a (single-key) secure circuit-constrained IPF.

By design, our circuit-constrained IPF provides two ways to invert: the "honest" method where on input (ct, y_2), the evaluator uses the PRF key $k^{(2)}$ to compute a (candidate) preimage $x \leftarrow \mathsf{F}_2(k^{(2)}, \mathsf{ct}) \oplus y_2$, and the "trapdoor" method where an evaluator who holds the decryption key for the public-key

encryption scheme simply decrypts ct to recover the (candidate) preimage x. The inversion trapdoor plays an important role in the security analysis of our circuit-constrained IPF because it enables the reduction algorithm to properly simulate the inversion oracle queries in the IPF security game. We refer to the full version of this paper [16] for the complete details.

Theorems 4.3 and 4.7 state that our puncturable IPF and circuit-constrained IPF are secure assuming the security (and privacy) of the underlying constrained PRFs (and in the latter case, CCA-security of the public-key encryption scheme). While it may seem that security of the IPF should directly follow from security of the underlying puncturable (or constrained) PRFs, several complications arise in the security analysis because we give the adversary access to an IPF inversion oracle in the security game. As a result, our security analysis requires a more intricate hybrid argument where we appeal to the security of the underlying constrained PRFs multiple times. We provide the complete proofs in the full version [16].

A multi-key constrained IPFs from $i\mathcal{O}$. In Sect. 7, we also show that an indistinguishability obfuscation of the puncturable IPF from (1.1) gives a multi-key circuit-constrained IPF. This construction parallels the Boneh-Zhandry construction of multi-key circuit-constrained PRFs from standard puncturable PRFs and indistinguishability obfuscation [20].

Supporting key-delegation. Several constrained PRF constructions support a mechanism called key-delegation [19,26,27], where the holder of a constrained PRF key can further constrain the key. For instance, the holder of a constrained key k_f for a function f can further constrain the key to a function of the form $f \wedge g$ where $(f \wedge g)(x) = 1$ if and only if $f(x) = g(x) = 1$. In Sect. 6, we describe how our circuit-constrained IPF can be extended to support key-delegation.

Open problems. Our impossibility results for constrained PRPs rule out any constraint class that enables evaluation on a non-negligible fraction of the domain. For example, this rules out the possibility of a puncturable PRP. Can we build constrained PRPs for constraint families that allow evaluation on a more restricted subset of the domain? For instance, do prefix-constrained PRPs exist?

Our circuit-constrained IPF from LWE is secure only if a *single* constrained key is issued. In Sect. 6, we show how to modify our construction to support giving out a pre-determined number of keys, provided that each successive key adds a further constraint on the previous key (i.e., via key delegation). Is there an IPF that supports multiple constrained keys for an arbitrary set of circuit constraints (and does not rely on strong assumptions such as $i\mathcal{O}$ or multilinear maps)? A positive answer would also give a circuit-constrained PRF that supports multiple keys, which is currently an open problem.

Our circuit-constrained IPF relies on the LWE assumption. Can we build constrained IPFs from one-way functions? For example, the tree-based PRF of [34] gives a prefix-constrained PRF from one-way functions. Can we build a prefix-constrained IPF from one-way functions?

1.2 Related Work

Authenticated encryption was first formalized over a sequence of works [10,11, 39,48,49]. Deterministic authenticated encryption, and the notion of a pseudo-random injection, were introduced in [50]. These notions have been further studied in [37,38]. Our circuit-constrained IPF relies on derandomizing a public-key encryption scheme. Similar techniques have been used in the context of constructing deterministic public-key encryption [6,7,12,31]. Note however that an IPF is a *secret-key* primitive, so in our setting, the randomness used for encryption can be derived using a PRF on the message rather than as a publicly-computable function on the input. This critical difference eliminates the need to make entropic assumptions on the inputs.

Since the introduction of constrained PRFs in [19,21,40], numerous works have studied constraining other cryptographic primitives such as verifiable random functions (VRFs) [26,27,29] and signatures [8,21]. Other works have focused on constructing adaptively-secure constrained PRFs [30,35,36] and constrained PRFs for inputs of unbounded length [27,28].

2 Preliminaries

For a positive integer n, we write $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For a distribution $\mathcal{D}$, we write $x \leftarrow \mathcal{D}$ to denote that x is sampled from $\mathcal{D}$; for a finite set S, we write $x \xleftarrow{R} S$ to denote that x is sampled uniformly from S. Throughout this work, we write λ for the security parameter. We say a function $f(\lambda)$ is negligible in λ if $f(\lambda) = o(1/\lambda^c)$ for all $c \in \mathbb{N}$. We denote this by writing $f(\lambda) = \mathrm{negl}(\lambda)$. We say that an algorithm is efficient if it runs in probabilistic polynomial time in the length of its input. We write $\mathrm{poly}(\lambda)$ to denote a quantity that is bounded by some polynomial in λ. We say that an event occurs with overwhelming probability if its complement occurs with negligible probability, and that it occurs with noticeable probability if it occurs with non-negligible probability. We say that two families of distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable if no efficient algorithm can distinguish between $\mathcal{D}_1$ and $\mathcal{D}_2$, except with negligible probability. We say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are statistically indistinguishable if the statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is negligible.

Function families. For two sets $\mathcal{X}$, $\mathcal{Y}$, we write $\mathsf{Funs}[\mathcal{X}, \mathcal{Y}]$ to denote the set of functions from $\mathcal{X}$ to $\mathcal{Y}$. We write $\mathsf{InjFuns}[\mathcal{X}, \mathcal{Y}]$ to denote the set of *injective* functions from $\mathcal{X}$ to $\mathcal{Y}$. For an injective function $f \in \mathsf{InjFuns}[\mathcal{X}, \mathcal{Y}]$, we denote by $f^{-1} : \mathcal{Y} \to \mathcal{X} \cup \{\perp\}$ the function where $f^{-1}(y) = x$ if $y = f(x)$, and $\perp$ if there is no such $x \in \mathcal{X}$. We sometimes refer to f^{-1} as the (generalized) inverse of f. When the domain and range are the same, the set $\mathsf{InjFuns}[\mathcal{X}, \mathcal{X}]$ is precisely the set of permutations on $\mathcal{X}$.

2.1 CCA-Secure Public-Key Encryption

A PKE scheme consists of three algorithms $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Encrypt}, \mathsf{PKE.Decrypt})$ over a message space $\mathcal{M}$ and a ciphertext space $\mathcal{T}$ with the following properties:

- $\mathsf{PKE.Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: On input the security parameter λ, the setup algorithm generates a public key pk and a secret key sk.
- $\mathsf{PKE.Encrypt}(\mathsf{pk}, m) \to \mathsf{ct}$: On input a public key pk and a message $m \in \mathcal{M}$, the encryption algorithm returns a ciphertext $\mathsf{ct} \in \mathcal{T}$.
- $\mathsf{PKE.Decrypt}(\mathsf{sk}, \mathsf{ct}) \to m$: On input a secret key sk and a ciphertext $\mathsf{ct} \in \mathcal{T}$, the decryption algorithm outputs a message $m \in \mathcal{M} \cup \{\perp\}$.

We say that a PKE scheme is correct if for all keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$, and for all messages $m \in \mathcal{M}$, we have that

$$\Pr[\mathsf{PKE.Decrypt}(\mathsf{sk}, \mathsf{PKE.Encrypt}(\mathsf{pk}, m)) = m] = 1.$$

Definition 2.1 (CCA-Security [43,46]). *Let* $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Encrypt}, \mathsf{PKE.Decrypt})$ *be a PKE scheme with message space* $\mathcal{M}$ *and ciphertext space* $\mathcal{T}$*, and let* $\mathcal{A}$ *be an efficient adversary. For a security parameter* λ *and a bit* $b \in \{0,1\}$*, we define the CCA-security experiment* $\mathsf{Expt}^{(\mathsf{CCA})}_{\mathcal{A},\mathsf{PKE}}(\lambda, b)$ *as follows. The challenger first samples* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$*. The adversary can then issue decryption oracle queries and up to one challenge oracle query.*[1] *Depending on the bit* $b \in \{0,1\}$*, the challenger responds to each query as follows:*

- ***Decryption oracle.*** *On input a ciphertext* $\mathsf{ct} \in \mathcal{T}$*, the challenger responds with the decryption* $m \leftarrow \mathsf{PKE.Decrypt}(\mathsf{sk}, \mathsf{ct})$*.*
- ***Challenge oracle.*** *On input two messages* $m_0, m_1 \in \mathcal{M}$*, the challenger responds with the ciphertext* $\mathsf{ct}^* \leftarrow \mathsf{PKE.Encrypt}(\mathsf{pk}, m_b)$*.*

At the end of the experiment, the adversary $\mathcal{A}$ *outputs a bit* $b' \in \{0,1\}$ *which is the output of the experiment. An adversary* $\mathcal{A}$ *is admissible if* $\mathcal{A}$ *does not submit the ciphertext* ct^* *it received from the challenge oracle to the decryption oracle. We say that* PKE *is secure against chosen-ciphertext attacks (CCA-secure) if for all efficient and admissible adversaries* $\mathcal{A}$*,*

$$\left|\Pr[\mathsf{Expt}^{(\mathsf{CCA})}_{\mathcal{A},\mathsf{PKE}}(\lambda, 0) = 1] - \Pr[\mathsf{Expt}^{(\mathsf{CCA})}_{\mathcal{A},\mathsf{PKE}}(\lambda, 1) = 1]\right| = \mathrm{negl}(\lambda).$$

[1] In the public-key setting, security against adversaries that make a single challenge query implies security against adversaries that make multiple challenge queries (via a standard hybrid argument).

Smoothness. In our security analysis, we require that our public-key encryption scheme satisfy an additional smoothness property. We say that a public-key encryption scheme is smooth if every message can encrypt to a super-polynomial number of potential ciphertexts. This property is satisfied by most natural public-key encryption schemes. After all, if the adversary can find a message m that has only polynomially-many ciphertexts, then the adversary can trivially break semantic security of the scheme. Of course, it is possible to craft public-key encryption schemes [9] where there exist (hard-to-find) messages that encrypt to only polynomially-many ciphertexts. We give the formal definition of smoothness in Definition 2.2.

Definition 2.2 (Smoothness [9, adapted]**).** *A* PKE *scheme* $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Encrypt}, \mathsf{PKE.Decrypt})$ *with message space* $\mathcal{M}$ *and ciphertext space* $\mathcal{T}$ *is* smooth *if for all messages* $m \in \mathcal{M}$ *and all strings* $\mathsf{ct} \in \mathcal{T}$*,*

$$\Pr\left[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda) : \mathsf{PKE.Encrypt}(\mathsf{pk}, m) = \mathsf{ct}\right] = \mathrm{negl}(\lambda),$$

where the probability is taken over the randomness in PKE.Setup *and* PKE.Encrypt.

3 Invertible PRFs

In this section, we introduce the notion of an invertible pseudorandom function (IPF). We then extend our notions to that of a *constrained* IPF. We begin by recalling the definition of a pseudorandom function (PRF) [34].

Definition 3.1 (Pseudorandom Function [34]**).** *A pseudorandom function (PRF) with key-space* $\mathcal{K}$*, domain* $\mathcal{X}$*, and range* $\mathcal{Y}$ *is a function* $\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *that can be computed by a deterministic polynomial-time algorithm. A PRF can also include a setup algorithm* $\mathsf{F.Setup}(1^\lambda)$ *that on input the security parameter* λ*, outputs a key* $k \in \mathcal{K}$*. A function* F *is a secure PRF if for all efficient adversaries* $\mathcal{A}$*,*

$$\left|\Pr\left[k \leftarrow \mathsf{F.Setup}(1^\lambda) : \mathcal{A}^{\mathsf{F}(k,\cdot)}(1^\lambda) = 1\right] - \Pr\left[\mathsf{R} \xleftarrow{R} \mathsf{Funs}[\mathcal{X}, \mathcal{Y}] : \mathcal{A}^{\mathsf{R}(\cdot)}(1^\lambda) = 1\right]\right| = \mathrm{negl}(\lambda).$$

An invertible pseudorandom function (IPF) is an injective PRF whose inverse function can be computed efficiently (given the secret key). This requirement that the inverse be efficiently computable is the key distinguishing factor between IPFs and injective PRFs. For instance, injective PRFs can be constructed by composing a sufficiently-expanding PRF with a pairwise-independent hash function. However, it is unclear how to invert such a PRF. We now give the definition of an IPF.

Definition 3.2 (Invertible Pseudorandom Functions). *An invertible pseudorandom function (IPF) with key-space* $\mathcal{K}$*, domain* $\mathcal{X}$*, and range* $\mathcal{Y}$ *consists of two functions* $\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *and* $\mathsf{F}^{-1}\colon \mathcal{K} \times \mathcal{Y} \to \mathcal{X} \cup \{\perp\}$*. An IPF can also include a setup algorithm* $\mathsf{F.Setup}(1^\lambda)$ *that on input the security parameter* λ*, outputs a key* $k \in \mathcal{K}$*. The functions* F *and* F^{-1} *satisfy the following properties:*

- *Both* F *and* F^{-1} *can be computed by deterministic polynomial-time algorithms.*
- *For all security parameters* λ *and all keys* k *output by* $\mathsf{F.Setup}(1^\lambda)$*, the function* $\mathsf{F}(k,\cdot)$ *is an injective function from* $\mathcal{X}$ *to* $\mathcal{Y}$*. Moreover, the function* $\mathsf{F}^{-1}(k,\cdot)$ *is the (generalized) inverse of* $\mathsf{F}(k,\cdot)$*.*

Definition 3.3 (Pseudorandomness). *An IPF* $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *is secure if for all efficient adversaries* $\mathcal{A}$*,*

$$\Big|\Pr\left[k \leftarrow \mathsf{F.Setup}(1^\lambda) : \mathcal{A}^{\mathsf{F}(k,\cdot),\,\mathsf{F}^{-1}(k,\cdot)}(1^\lambda)\right] - \Pr\left[\mathsf{R} \xleftarrow{R} \mathsf{InjFuns}[\mathcal{X},\mathcal{Y}] : \mathcal{A}^{\mathsf{R}(\cdot),\,\mathsf{R}^{-1}(\cdot)}(1^\lambda)\right]\Big| = \mathrm{negl}(\lambda).$$

Remark 3.4 (Strong vs. Weak Pseudorandomness). The pseudorandomness requirement for an IPF (Definition 3.3) requires that the outputs of an IPF be indistinguishable from random against adversaries that can query the IPF in both the forward direction as well as the backward direction. We can also consider a weaker notion of pseudorandomness where the adversary is given access to an evaluation oracle $\mathsf{F}(k,\cdot)$, but not an inversion oracle $\mathsf{F}^{-1}(k,\cdot)$. Motivated by the applications we have in mind, in this work, we focus exclusively on building IPFs satisfying the *strong* notion of pseudorandomness from Definition 3.3, where the adversary can evaluate the IPF in both directions.

3.1 Constrained PRFs and IPFs

We next review the notion of a constrained PRF [19,21,40] and then extend these definitions to constrained IPFs.

Definition 3.5 (Constrained PRF [19,21,40]). *A PRF* $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *is said to be constrained with respect to a predicate family* $\mathcal{F} = \{f : \mathcal{X} \to \{0,1\}\}$ *if there are two additional algorithms* $(\mathsf{F.Constrain}, \mathsf{F.Eval})$ *with the following properties:*

- $\mathsf{F.Constrain}(k, f) \to k_f$*: On input a PRF key* $k \in \mathcal{K}$ *and a function* $f \in \mathcal{F}$*, the constraining algorithm outputs a constrained key* k_f*.*
- $\mathsf{F.Eval}(k_f, x) \to y$*: On input a constrained key* k_f *and a point* $x \in \mathcal{X}$*, the evaluation algorithm outputs a value* $y \in \mathcal{Y}$*.*

We say that a constrained PRF is correct for a function family $\mathcal{F}$ *if for all* $k \leftarrow \mathsf{F.Setup}(1^\lambda)$*, every function* $f \in \mathcal{F}$*, and every input* $x \in \mathcal{X}$ *where* $f(x) = 1$*, we have that*

$$\mathsf{F.Eval}(\mathsf{F.Constrain}(k, f), x) = \mathsf{F}(k, x).$$

Definition 3.6 (Constrained PRF Security Experiment). *Let* $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *be a constrained PRF with respect to a function family* $\mathcal{F}$*, and let* $\mathcal{A}$ *be an efficient adversary. In the constrained PRF security experiment* $\mathsf{Expt}^{(\mathsf{PRF})}_{\mathcal{A},\mathsf{F}}(\lambda, b)$ *(parameterized by a security parameter* λ *and a bit* $b \in \{0,1\}$*), the challenger begins by sampling a key* $k \leftarrow \mathsf{F.Setup}(1^\lambda)$ *and a random function* $\mathsf{R} \xleftarrow{R} \mathsf{Funs}[\mathcal{X},\mathcal{Y}]$*. The adversary is allowed to make constrain, evaluation, and challenge oracle queries. Depending on the value of the bit* $b \in \{0,1\}$*, the challenger responds to each oracle query as follows:*

- ***Constrain oracle.*** *On input a function $f \in \mathcal{F}$, the challenger responds with a constrained key $k_f \leftarrow \mathsf{F.Constrain}(k, f)$.*
- ***Evaluation oracle.*** *On input a point $x \in \mathcal{X}$, the challenger returns $y = \mathsf{F}(k, x)$.*
- ***Challenge oracle.*** *On input a point $x \in \mathcal{X}$, the challenger returns $y = \mathsf{F}(k, x)$ to $\mathcal{A}$ if $b = 0$ and $y = \mathsf{R}(x)$ if $b = 1$.*

Finally, at the end of the experiment, the adversary $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$ which is also the output of the experiment.

Definition 3.7 (Constrained PRF Security). *Let $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a constrained PRF for a function family $\mathcal{F}$. We say that an adversary $\mathcal{A}$ is admissible for the constrained PRF security experiment (Definition 3.6) if the following conditions hold:*

- *For all constrain queries $f \in \mathcal{F}$ and challenge queries $x^* \in \mathcal{X}$ the adversary makes, $f(x^*) = 0$.*
- *For all evaluation queries $x \in \mathcal{X}$ and challenge queries $x^* \in \mathcal{X}$ the adversary makes, $x \neq x^*$.*

We say that F is a secure constrained PRF if for all efficient and admissible adversaries $\mathcal{A}$,

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

Without loss of generality, we restrict the adversary to make at most one challenge query in the constrained PRF security experiment.[2]

Remark 3.8 (Selective vs. Adaptive Security). The constrained PRF security game (Definition 3.6) allows the adversary to *adaptively* choose the challenge point after making constrain and evaluation queries. We can also define a *selective* notion of security where the adversary must commit to its challenge query at the beginning of the security game (before it starts making queries). Using a standard technique called *complexity leveraging* [13], selective security implies adaptive security at the expense of a super-polynomial loss in the security reduction. For instance, this is the technique used in [19] in the context of constrained PRFs.

Remark 3.9 (Single-Key Security). Brakerski and Vaikuntanathan [23] considered another relaxation of Definition 3.7 where in the constrained PRF security game (Definition 3.6), the adversary is restricted to making a single query to the constrain oracle. In the single-key setting, we can consider the notion of *selective-function* security, where the adversary must commit to its constrain oracle query at the beginning of the security experiment. Thus, in this setting, there are two different notions of selectivity: the usual notion where the adversary commits

[2] As noted in [19], a standard hybrid argument shows that security against adversaries making a single challenge query implies security against adversaries making multiple challenge queries.

to the challenge point (Remark 3.8) and selective-function security where the adversary commits to the function. Many of the lattice-based (single-key) constrained PRF constructions [15,22–24] are selectively secure in the choice of the constraint function, but adaptively secure in the choice of the challenge point.

Definition 3.10 (Constrained IPF). *An IPF* $(\mathsf{F}, \mathsf{F}^{-1})$ *with key-space* $\mathcal{K}$*, domain* $\mathcal{X}$*, and range* $\mathcal{Y}$ *is said to be constrained with respect to a function family* $\mathcal{F} = \{f : \mathcal{X} \to \{0,1\}\}$ *if there are three additional algorithms* $(\mathsf{F.Constrain}, \mathsf{F.Eval}, \mathsf{F.Eval}^{-1})$ *with the following properties:*

- $\mathsf{F.Constrain}(k, f) \to k_f$*: On input a PRF key* $k \in \mathcal{K}$ *and a function* $f \in \mathcal{F}$*, the constraining algorithm outputs a constrained key* k_f*.*
- $\mathsf{F.Eval}(k_f, x) \to y$*: On input a constrained key* k_f *and a value* $x \in \mathcal{X}$*, the evaluation algorithm outputs a value* $y \in \mathcal{Y}$*.*
- $\mathsf{F.Eval}^{-1}(k_f, y) \to x$*: On input a constrained key* k_f *and a value* $y \in \mathcal{Y}$*, the evaluation algorithm outputs a value* $x \in \mathcal{X} \cup \{\perp\}$*.*

We say that a constrained IPF is correct for a function family $\mathcal{F}$ *if for all keys* $k \leftarrow \mathsf{F.Setup}(1^\lambda)$*, every function* $f \in \mathcal{F}$*, and* $k_f \leftarrow \mathsf{F.Constrain}(k, f)$*, the following two properties hold:*

- *For all inputs* $x \in \mathcal{X}$ *where* $f(x) = 1$*,* $\mathsf{F.Eval}(k_f, x) = \mathsf{F}(k, x)$*.*
- *For all inputs* $y \in \mathcal{Y}$ *where there exists* $x \in \mathcal{X}$ *such that* $\mathsf{F}(k, x) = y$ *and* $f(x) = 1$*, then* $\mathsf{F.Eval}^{-1}(k_f, y) = \mathsf{F}^{-1}(k, y)$*.*

Definition 3.11 (Constrained IPF Security Experiment). *Let* $(\mathsf{F}, \mathsf{F}^{-1})$ *be an IPF with key-space* $\mathcal{K}$*, domain* $\mathcal{X}$*, range* $\mathcal{Y}$*, and constrained with respect to a function family* $\mathcal{F}$*. Let* $\mathcal{A}$ *be an efficient adversary. The constrained IPF security experiment* $\mathsf{Expt}^{(\mathsf{IPF})}_{\mathcal{A},\mathsf{F}}(\lambda, b)$ *is defined exactly as the constrained PRF security experiment* $\mathsf{Expt}^{(\mathsf{PRF})}_{\mathcal{A},\mathsf{F}}(\lambda, b)$ *(except with the IPF in place of the PRF and the random function* R *is sampled from* $\mathsf{InjFuns}[\mathcal{X}, \mathcal{Y}]$*), and in addition to the constrain, evaluation, and challenge oracles, the adversary is also given access to an inversion oracle:*

- ***Inversion oracle.*** *On input a point* $y \in \mathcal{Y}$*, the challenger returns* $\mathsf{F}^{-1}(k, y)$*.*

At the end of the experiment, the adversary $\mathcal{A}$ *outputs a bit* $b' \in \{0,1\}$*, which is the output of the experiment.*

Definition 3.12 (Constrained IPF Security). *Let* $(\mathsf{F}, \mathsf{F}^{-1})$ *be an IPF with key-space* $\mathcal{K}$*, domain* $\mathcal{X}$*, range* $\mathcal{Y}$*, and constrained with respect to a function family* $\mathcal{F}$*. We say that an adversary* $\mathcal{A}$ *is admissible for the constrained IPF security experiment (Definition 3.11) if the following conditions hold:*

- *For all constrain queries* $f \in \mathcal{F}$ *and challenge queries* $x^* \in \mathcal{X}$ *the adversary makes,* $f(x^*) = 0$*.*
- *For all evaluation queries* $x \in \mathcal{X}$ *and challenge queries* $x^* \in \mathcal{X}$ *the adversary makes,* $x \neq x^*$*.*

– *For all inversion queries $y \in \mathcal{Y}$ the adversary makes, $y \notin \mathcal{Y}^*$, where $\mathcal{Y}^*$ is the set of responses to the adversary's challenge oracle queries from the challenger.*

We say that F *is a secure constrained IPF if for all efficient and admissible adversaries* $\mathcal{A}$*,*

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

As in Definition 3.7, we restrict the adversary to making at most one challenge query in the constrained IPF security experiment.

Remark 3.13 (Selective vs. Adaptive Security for IPFs). As with constrained PRFs, we can define a notion of selective security for IPFs, where the adversary commits to its challenge query at the beginning of the constrained IPF security experiment (Remark 3.8). Similarly, we can consider a single-key variant of the security game, where the adversary makes a single constrain oracle query. In this case, we can also define the corresponding notion of selective-function security (Remark 3.9).

Puncturable PRFs and IPFs. An important subclass of constrained PRFs is the class of punctured PRFs [19,21,40]. A punctured PRF over a domain $\mathcal{X}$ is a PRF constrained with respect to the family of point functions: $\mathcal{F} = \{f_{x^*} : \mathcal{X} \to \{0,1\} \mid x^* \in \mathcal{X}\}$, where $f_{x^*}(x) = 1$ for all $x \neq x^*$ and $f_{x^*}(x^*) = 0$. For notational convenience, when working with a puncturable PRF $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, we replace the $\mathsf{F.Constrain}$ algorithm with the $\mathsf{F.Puncture}$ algorithm that takes as input a PRF key k and a point $x^* \in \mathcal{X}$ and outputs a punctured key k_{x^*} (a key constrained to the point function f_{x^*}). We extend these notions accordingly to puncturable IPFs.

3.2 Private Constrained PRFs

One of the key primitives we will need to build constrained IPFs is a *private* constrained PRF [18]. A *private* constrained PRF is a constrained PRF with the additional property that the constrained keys hide the underlying constraining function. Boneh et al. [18] showed how to construct private constrained PRFs for all circuits using indistinguishability obfuscation. Recently, a number of works have shown how to construct private constrained PRFs for puncturing constraints [15], NC^1 constraints [24], and general circuit constraints [22] from standard lattice assumptions. We now review the simulation-based notion of privacy considered in [15,24].

Definition 3.14 (Single-Key Constraint Privacy *[15,24]***).** *Let* $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *be a constrained PRF with respect to a function family* $\mathcal{F}$*. We say that* F *is a single-key, selectively-private constrained PRF for* $\mathcal{F}$ *if for all efficient adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a stateful simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ *such that the following two distributions are computationally indistinguishable:*

Experiment $\mathsf{Real}_{\mathcal{A},\mathsf{F}}(\lambda)$*:*

- $(f, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(1^\lambda)$
- $k \leftarrow \mathsf{F.Setup}(1^\lambda)$
- $k_f \leftarrow \mathsf{F.Constrain}(k, f)$
- $b \leftarrow \mathcal{A}^{\mathsf{F}(k,\cdot)}(k_f, \mathsf{st}_{\mathcal{A}})$
- *Output* b

Experiment $\mathsf{Ideal}_{\mathcal{A},\mathcal{S},\mathsf{F}}(\lambda)$*:*

- $(f, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(1^\lambda)$
- $(k_f, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_1(1^\lambda)$
- $b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Eval}}(\cdot)}(k_f, \mathsf{st}_{\mathcal{A}})$*, where the ideal evaluation oracle* $\mathcal{O}_{\mathsf{Eval}}(\cdot)$ *takes as input a point* $x \in \mathcal{X}$*, computes* $(y, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_2(x, f(x), \mathsf{st}_{\mathcal{S}})$*, and returns* y
- *Output* b

Observe that the simulator (S_1, S_2) in the ideal experiment is not given the function f as input. Nevertheless, the simulator can simulate k_f as in the real experiment. This implies that the adversary learns nothing about f from k_f beyond the value of f at points $x \in \mathcal{X}$ where the adversary asks for $\mathsf{F}(k, x)$. Leaking this minimal information about f is unavoidable.

3.3 Special Cases: PRPs and Constrained PRPs

Invertible pseudorandom functions can be viewed as a generalization of pseudorandom permutations (PRPs) where we allow the range of the function to be larger than its domain. A PRP is an IPF where the domain and range are identical. Our definitions for constrained IPFs can be similarly adapted to the setting of constrained PRPs. In this section, we make several observations on the (non)-existence of constrained PRPs, as well as discuss some possible relaxations of the security requirements to circumvent the impossibility results. We first show that constrained PRPs (for any family of constraints) on polynomial-size domains do not exist. Next, we show that even over large domains, security for many natural classes of constraints, including puncturing, is impossible to achieve. Our argument here can be extended to derive a lower bound on the size of the range of any IPF that supports puncturing constraints (or more generally, any constraint that enables evaluation a non-negligible fraction of the domain).

Remark 3.15 (Small-Domain Constrained PRPs are Insecure). No constrained PRP over a polynomial-size domain can be secure under the standard pseudorandomness definition of Definition 3.12. This follows from the fact that a PRP is easily distinguishable from a PRF when the domain is small—given even a single input-output pair (x^*, y^*) for the PRP, the adversary already learns something about the values of the PRP at any point $x \neq x^*$ (namely, the value of the PRP at x cannot be y^*). Thus, the adversary can distinguish the real output of the PRP at $x \neq x^*$ (which cannot be y^*) from a uniformly random value (which can be y^* with noticeable probability when the domain is small).

Theorem 3.16 (Limitations on Constrained PRPs). *Let* $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ *be a PRP constrained with respect to a predicate family* $\mathcal{F}$*. For each predicate*

$f \in \mathcal{F}$, let $S_f = \{x \in \mathcal{X} : f(x) = 1\}$ denote the set of allowable points for f. If there exists $f \in \mathcal{F}$ where the quantity $|S_f| / |\mathcal{X}|$ is non-negligible, then F cannot be secure in the sense of Definition 3.12.

Proof. Suppose there exists $f \in \mathcal{F}$ where $|S_f| / |\mathcal{X}|$ is non-negligible. We construct the following adversary for the constrained security game:

1. First, $\mathcal{A}$ makes a constrain query for f and a challenge query on an arbitrary $x^* \in \mathcal{X}$ where $f(x^*) = 0$. It receives from the challenger a punctured key k_f and a challenge value y^*.
2. Then, $\mathcal{A}$ computes $x \leftarrow \mathsf{F.Eval}^{-1}(k_f, y^*)$, and outputs 1 if either of the following conditions hold:
   - if $f(x) = 0$, or
   - if $\mathsf{F.Eval}(k_f, x) \neq y^*$.

   Otherwise, $\mathcal{A}$ outputs 0.

To complete the analysis, we compute the probability that $\mathcal{A}$ outputs 1:

- Suppose $y^* = \mathsf{F}(k, x^*)$. Consider the case where $f(x) = 1$. Note in particular that this means $x \neq x^*$. By correctness of F, we have that $\mathsf{F.Eval}(k_f, x) = \mathsf{F}(k, x)$. Moreover, since $\mathsf{F}(k, \cdot)$ is a permutation, it follows that $\mathsf{F}(k, x) \neq \mathsf{F}(k, x^*) = y^*$. Thus, in this case, either $f(x) = 0$ or $\mathsf{F.Eval}(k_f, x) \neq y^*$, so we conclude that $\mathcal{A}$ outputs 1 with probability 1.
- Suppose y^* is uniformly random over $\mathcal{X}$. Let $\hat{x} = \mathsf{F}^{-1}(k, y^*)$. Suppose that $f(\hat{x}) = 1$. Then, by correctness of F, we have that

$$x = \mathsf{F.Eval}^{-1}(k_f, y^*) = \mathsf{F}^{-1}(k, y^*) = \hat{x}.$$

  Moreover, since $f(\hat{x}) = 1$, we have

$$\mathsf{F.Eval}(k_f, x) = \mathsf{F.Eval}(k_f, \hat{x}) = \mathsf{F}(k, \hat{x}) = y^*.$$

  Thus, whenever $f(\hat{x}) = 1$, adversary $\mathcal{A}$ outputs 1 with probability 0. Since y^* is uniformly random over $\mathcal{X}$ and $\mathsf{F}(k, \cdot)$ is a permutation,

$$\Pr[\mathcal{A} \text{ outputs } 1] \leq \Pr[f(\hat{x}) = 0] = 1 - |S_f| / |\mathcal{X}|.$$

We conclude that $\mathcal{A}$ breaks the constrained security of F with advantage $|S_f| / |\mathcal{X}|$, which is non-negligible by assumption. □

Corollary 3.17 (Puncturable PRPs are Insecure). *Let $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ be a puncturable PRP. Then, F is insecure in the sense of Definition 3.12.*

Proof. The set of allowable points S_f for a puncturing constraint f is always $|\mathcal{X}| - 1$, so the ratio $|S_f| / |\mathcal{X}|$ is always non-negligible. The claim then follows from Theorem 3.16. □

Remark 3.18 (Constrained PRPs for Very Restricted Constraint Classes). Theorem 3.16 rules out any constrained PRP that supports issuing constrained keys that can be used to evaluate on a non-negligible fraction of the domain. It does leave open the possibility of building constrained PRPs where each constrained key can only be used to evaluate on a *negligible* fraction of the domain. A natural class of constraints that satisfies this property is the class of *prefix-constrained PRPs* (for a prefix of super-logarithmic size). We leave it as an open problem to construct a prefix-constrained PRP, or more generally, a constrained PRP where all of the constrained keys can only be used to evaluate on a negligible fraction of the domain.

Remark 3.19 (Constrained IPFs Must be Expanding). The attack from the proof of Theorem 3.16 also extends to the setting where $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a constrained *IPF* with a small range. Specifically, if $|\mathcal{Y}| \leq |\mathcal{X}| \cdot \mathrm{poly}(\lambda)$, and F supports issuing a constrained key for a function $f : \mathcal{X} \to \{0,1\}$ where $|S_f| / |\mathcal{X}|$ is non-negligible, then F cannot be secure in the sense of Definition 3.12. In this setting, we would modify the distinguisher in the proof of Theorem 3.16 to additionally output 1 if $x = \perp$. With this modification, the distinguishing advantage of the attack only decreases by a polynomial factor $|\mathcal{X}| / |\mathcal{Y}| = 1/\mathrm{poly}(\lambda)$. Therefore, any constrained IPF that admits a constraint that can be used to evaluate the IPF on a non-negligible fraction of the domain must necessarily have a range that is larger than the domain by at least a super-polynomial factor. Concretely, a puncturable IPF must have a range that is super-polynomially larger than the domain.

Remark 3.20 (Weaker Security Relations). The lower bound in Theorem 3.16 only applies when we require that the IPF value at a constrained point appear pseudorandom given the constrained key. One way to circumvent the lower bound is to consider a weaker security notion where we just require the IPF value at a constrained point to be *unpredictable* rather than pseudorandom (given the constrained key). In other words, no efficient adversary should be able to predict $\mathsf{F}(k, x)$ given a constrained key k_f that does not allow evaluation at x. While the weaker security properties are potentially satisfiable, they may not be sufficient for specific applications.

4 Constructing Constrained IPFs

We now turn to constructing constrained IPFs and give two main constructions in this section. Our main constructions use private constrained (non-invertible) PRFs as the primary tool. As a warm-up, we first construct a puncturable IPF from a private puncturable PRF in Sect. 4.1. We then show how the basic IPF construction can be extended to obtain a (single-key) circuit-constrained IPF in Sect. 4.2. In Sect. 7, we also show that an indistinguishability obfuscation of the basic puncturable IPF gives a *multi-key* circuit-constrained IPF.

4.1 Warm-Up: Puncturable IPF from Private Puncturable PRFs

We begin by showing how to construct a puncturable IPF on a domain $\mathcal{X}$ from a private puncturable PRF on $\mathcal{X}$. We describe the construction and then show in Theorems 4.2 and 4.3 that it is a secure puncturable IPF.

Construction 4.1. *Fix a domain $\mathcal{X} = \{0,1\}^n$ where $n = n(\lambda)$. Let $\mathsf{F}_1 \colon \mathcal{K}_1 \times \mathcal{X} \to \mathcal{V}$ be an injective puncturable PRF with key-space $\mathcal{K}_1$ and range $\mathcal{V}$. Let $\mathsf{F}_2 \colon \mathcal{K}_2 \times \mathcal{V} \to \mathcal{X}$ be a private puncturable PRF with key-space $\mathcal{K}_2$. The puncturable IPF $\mathsf{F} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ with key-space $\mathcal{K} = \mathcal{K}_1 \times \mathcal{K}_2$, domain $\mathcal{X}$, and range $\mathcal{Y} = \mathcal{V} \times \mathcal{X}$ is defined as follows:*

- *The IPF key is a pair of keys $k = (k^{(1)}, k^{(2)}) \in \mathcal{K}_1 \times \mathcal{K}_2$ for the puncturable PRFs F_1 and F_2.*
- *On input $k = (k^{(1)}, k^{(2)}) \in \mathcal{K}_1 \times \mathcal{K}_2 = \mathcal{K}$, and $x \in \mathcal{X}$ the IPF is defined as the pair*

$$\mathsf{F}\big((k^{(1)}, k^{(2)}),\ x\big) := \Big(\mathsf{F}_1(k^{(1)}, x),\quad x \oplus \mathsf{F}_2(k^{(2)}, \mathsf{F}_1(k^{(1)}, x)\Big).$$

- *On input $k = (k^{(1)}, k^{(2)}) \in \mathcal{K}_1 \times \mathcal{K}_2 = \mathcal{K}$, and $y = (y_1, y_2) \in \mathcal{V} \times \mathcal{X} = \mathcal{Y}$, the inversion algorithm $\mathsf{F}^{-1}(k, y)$ first computes $x \leftarrow \mathsf{F}_2(k^{(2)}, y_1) \oplus y_2$ and outputs*

$$\mathsf{F}^{-1}(k, (y_1, y_2)) := \begin{cases} x & \text{if } y_1 = \mathsf{F}_1(k^{(1)}, x) \\ \perp & \text{otherwise.} \end{cases}$$

Next, we define the setup and constraining algorithms for $(\mathsf{F}, \mathsf{F}^{-1})$.

- $\mathsf{F}.\mathsf{Setup}(1^\lambda)$*: On input the security parameter λ, the setup algorithm samples two puncturable PRF keys $k^{(1)} \leftarrow \mathsf{F}_1.\mathsf{Setup}(1^\lambda)$ and $k^{(2)} \leftarrow \mathsf{F}_2.\mathsf{Setup}(1^\lambda)$. The setup algorithm outputs the IPF key $k = (k^{(1)}, k^{(2)})$.*
- $\mathsf{F}.\mathsf{Puncture}(k, x^*)$*: On input the IPF key $k = (k^{(1)}, k^{(2)})$ and a point $x^* \in \mathcal{X}$ to be punctured, the puncturing algorithm first computes $v^* \leftarrow \mathsf{F}_1(k^{(1)}, x^*)$. It then generates two punctured keys $k^{(1)}_{x^*} \leftarrow \mathsf{F}_1.\mathsf{Puncture}(k^{(1)}, x^*)$ and $k^{(2)}_{v^*} \leftarrow \mathsf{F}_2.\mathsf{Puncture}(k^{(2)}, v^*)$ and returns $k_{x^*} = \big(k^{(1)}_{x^*}, k^{(2)}_{v^*}\big)$.*
- $\mathsf{F}.\mathsf{Eval}(k_{x^*}, x)$*: On input the punctured key $k_{x^*} = (k^{(1)}_{x^*}, k^{(2)}_{v^*})$ and a point $x \in \mathcal{X}$, the evaluation algorithm first computes $y_1 \leftarrow \mathsf{F}_1.\mathsf{Eval}(k^{(1)}_{x^*}, x)$ and returns $y = (y_1, \mathsf{F}_2.\mathsf{Eval}(k^{(2)}_{v^*}, y_1) \oplus x)$.*
- $\mathsf{F}.\mathsf{Eval}^{-1}(k_{x^*}, y)$*: On input the punctured key $k_{x^*} = (k^{(1)}_{x^*}, k^{(2)}_{v^*})$, and $y = (y_1, y_2) \in \mathcal{V} \times \mathcal{X} = \mathcal{Y}$, the inversion algorithm begins by computing the quantity $x \leftarrow \mathsf{F}_2.\mathsf{Eval}(k^{(2)}_{v^*}, y_1) \oplus y_2$. It returns x if $\mathsf{F}_1.\mathsf{Eval}(k^{(1)}_{x^*}, x) = y_1$ and $\perp$ otherwise.*

We now state our correctness and security theorems. We provide the formal proofs in the full version [16].

Theorem 4.2. *Suppose F_1 is an injective puncturable PRF and F_2 is a puncturable PRF. Then, the IPF $(\mathsf{F}, \mathsf{F}^{-1})$ from Construction 4.1 is correct.*

Theorem 4.3. *Suppose* F_1 *is a selectively-secure puncturable PRF,* F_2 *is a selectively-secure, private puncturable PRF, and* $|\mathcal{X}|/|\mathcal{V}| = \mathrm{negl}(\lambda)$. *Then* $(\mathsf{F}, \mathsf{F}^{-1})$ *from Construction 4.1 is a selectively-secure puncturable IPF.*

Remark 4.4 (Adaptive Security). Theorem 4.3 shows that if the underlying puncturable PRFs in Construction 4.1 are selectively secure, then the resulting IPF is selectively secure. We note that if we instantiate the underlying PRFs with an adaptively-secure (private) puncturable PRF (for instance, the construction due to Canetti and Chen [24]), then the resulting IPF can also be shown to be adaptively secure (following a similar argument as that used in the proof of Theorem 4.3).

4.2 Circuit-Constrained IPF from Private Circuit-Constrained PRFs

In this section, we show how to extend our puncturable IPF construction from Sect. 4.1 to obtain a (single-key) constrained IPF for arbitrary circuit constraints. Our security analysis for our circuit-constrained IPF construction relies critically on the assumption that one of the underlying PRFs is a circuit-constrained PRF satisfying a strong simulation-based notion of privacy (Definition 3.14). Canetti and Chen [24] previously showed that even a 2-key private constrained PRF satisfying this simulation-based notion of privacy implies virtual black-box (VBB) obfuscation for the same underlying circuit class. Since VBB obfuscation for all circuits is impossible in the standard model [5], our construction is instantiatable only in the single-key setting, and thus, we present our construction in the single-key setting.

Construction 4.5. *Fix a domain* $\mathcal{X} = \{0,1\}^n$ *where* $n = n(\lambda)$. *Our circuit-constrained IPF construction for* NC^1 *(resp.,* P/poly*) relies on several primitives:*

- *Let* $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Encrypt}, \mathsf{PKE.Decrypt})$ *be a PKE scheme with message space* $\mathcal{X}$*, ciphertext space* $\mathcal{T}$*, and whose decryption function can be computed in* NC^1 *(resp.,* P/poly*). Let* $\mathcal{PK}$ *and* $\mathcal{SK}$ *denote the space of public keys and the space of secret keys, respectively, for* PKE. *Let* $\mathcal{V}$ *denote the space from which the randomness for encryption is sampled.*
- *Let* $\mathsf{F}_1 : \mathcal{K}_1 \times \mathcal{X} \to \mathcal{V}$ *be a circuit-constrained PRF for* NC^1 *(resp.,* P/poly*).*
- *Let* $\mathsf{F}_2 : \mathcal{K}_2 \times \mathcal{T} \to \mathcal{X}$ *be a private circuit-constrained PRF for* NC^1 *(resp.,* P/poly*).*[3]

[3] To simplify the presentation, we implicitly assume that the PRFs F_1 and F_2 support general circuit constraints (i.e., NC^1 constraints or P/poly constraints). However, we can also instantiate our construction using private constrained PRFs for weaker constraint classes, provided that the constraint class is expressive enough to include the decryption algorithm for a CCA-secure public-key encryption scheme (see Remark 4.8).

The constrained IPF $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *with key-space* $\mathcal{K} = \mathcal{K}_1 \times \mathcal{K}_2 \times \mathcal{PK} \times \mathcal{SK}$, *domain* $\mathcal{X}$, *and range* $\mathcal{Y} \subseteq \mathcal{T} \times \mathcal{X}$ *is defined as follows:*

- *The IPF key consists of two PRF keys* $(k^{(1)}, k^{(2)}) \in \mathcal{K}_1 \times \mathcal{K}_2$ *for* F_1 *and* F_2, *respectively, and a public/secret key-pair* $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{PK} \times \mathcal{SK}$ *for the public-key encryption scheme* PKE.
- *On input a key* $k = (k^{(1)}, k^{(2)}, \mathsf{pk}, \mathsf{sk}) \in \mathcal{K}$, *and* $x \in \mathcal{X}$, *the IPF* $\mathsf{F}(k, x)$ *computes randomness* $r_x \leftarrow \mathsf{F}_1(k^{(1)}, x)$, *a ciphertext* $\mathsf{ct} \leftarrow \mathsf{PKE.Encrypt}(\mathsf{pk}, x; r_x)$, *and outputs*

$$\mathsf{F}(k, x) := \Big(\mathsf{ct},\ \ \mathsf{F}_2(k^{(2)}, \mathsf{ct}) \oplus x\Big).$$

  Note that the public key pk *can also be included as part of the public parameters for the IPF.*
- *On input a key* $k = (k^{(1)}, k^{(2)}, \mathsf{pk}, \mathsf{sk}) \in \mathcal{K}$, *and* $(y_1, y_2) \in \mathcal{Y}$, *the inversion function* $\mathsf{F}^{-1}(k, (y_1, y_2))$ *first computes* $x \leftarrow \mathsf{F}_2(k^{(2)}, y_1) \oplus y_2$ *and* $r_x \leftarrow \mathsf{F}_1(k^{(1)}, x)$. *Finally, it outputs*

$$\mathsf{F}^{-1}(k, (y_1, y_2)) := \begin{cases} x & \text{if } y_1 = \mathsf{PKE.Encrypt}(\mathsf{pk}, x; r_x) \\ \perp & \text{otherwise.} \end{cases}$$

- *The range of the IPF* $\mathcal{Y}$ *is defined to be the space* $\mathcal{T}' \times \mathcal{X}$ *where* $\mathcal{T}' = \{\mathsf{PKE.Encrypt}(\mathsf{pk}, x; r)\}_{x \in \mathcal{X}, r \in \mathcal{V}}$ *is the subset of ciphertexts that correspond to a valid encryption of some message under the public key* pk.

Next, we define the setup and constraining algorithms for $(\mathsf{F}, \mathsf{F}^{-1})$.

- $\mathsf{F.Setup}(1^\lambda)$*: On input the security parameter* λ, *the setup algorithm samples two PRF keys* $k^{(1)} \leftarrow \mathsf{F}_1.\mathsf{Setup}(1^\lambda)$, $k^{(2)} \leftarrow \mathsf{F}_2.\mathsf{Setup}(1^\lambda)$, *and a public/secret key-pair for the PKE scheme:* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$. *It outputs the IPF key* $k = (k^{(1)}, k^{(2)}, \mathsf{pk}, \mathsf{sk})$.
- $\mathsf{F.Constrain}(k, f)$*: On input the IPF key* $k = (k^{(1)}, k^{(2)}, \mathsf{pk}, \mathsf{sk})$ *and a constraint function* $f \in \mathcal{F}$, *the algorithm first constrains* $k_f^{(1)} \leftarrow \mathsf{F}_1.\mathsf{Constrain}(k^{(1)}, f)$. *Then, it defines the function* $F_{\mathsf{sk},f} : \mathcal{T} \to \{0,1\}$ *as follows:*

$$F_{\mathsf{sk},f}(\mathsf{ct}) := \begin{cases} 1 & \text{if } \mathsf{PKE.Decrypt}(\mathsf{sk}, \mathsf{ct}) \neq \perp \text{ and } f(\mathsf{PKE.Decrypt}(\mathsf{sk}, \mathsf{ct})) = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

  The constrain algorithm constrains the key $k^{(2)}$ *to* $F_{\mathsf{sk},f}$ *and obtains* $k_F^{(2)} \leftarrow \mathsf{F}_2.\mathsf{Constrain}(k^{(2)}, F_{\mathsf{sk},f})$. *It then defines and returns the constrained key* $k_f = (k_f^{(1)}, k_F^{(2)}, \mathsf{pk})$. *Note that if* $\mathsf{PKE.Decrypt}(\mathsf{sk}, \cdot)$ *can be computed in* NC^1 *(resp.,* $\mathsf{P/poly}$*), then the function* $F_{\mathsf{sk},f}$ *can also be computed in* NC^1 *(resp.,* $\mathsf{P/poly}$*).*
- $\mathsf{F.Eval}(k_f, x)$*: On input the constrained key* $k_f = (k_f^{(1)}, k_F^{(2)}, \mathsf{pk})$, *and a point* $x \in \mathcal{X}$, *the algorithm first computes* $r_x \leftarrow \mathsf{F}_1.\mathsf{Eval}(k_f^{(1)}, x)$. *Then, it encrypts* $\mathsf{ct} \leftarrow \mathsf{PKE.Encrypt}(\mathsf{pk}, x; r_x)$ *and returns the tuple* $y = \Big(\mathsf{ct},\ \mathsf{F}_2.\mathsf{Eval}(k_F^{(2)}, \mathsf{ct}) \oplus x\Big)$.

– $\mathsf{F}.\mathsf{Eval}^{-1}(k_f, y)$: *On input the constrained key* $k_f = (k_f^{(1)}, k_F^{(2)}, \mathsf{pk})$*, and a point* $y = (y_1, y_2) \in \mathcal{Y}$*, the algorithm first computes* $x \leftarrow \mathsf{F}_2.\mathsf{Eval}(k_F^{(2)}, y_1) \oplus y_2$*. Then, it computes* $r_x \leftarrow \mathsf{F}_1.\mathsf{Eval}(k_f^{(1)}, x)$ *and* $\mathsf{ct} \leftarrow \mathsf{PKE}.\mathsf{Encrypt}(\mathsf{pk}, x; r_x)$*. If* $y_1 = \mathsf{ct}$*, then the algorithm returns* x*. Otherwise, it returns* $\perp$*.*

We now state our correctness and security theorems. We provide the formal proofs in the full version [16].

Theorem 4.6. *Suppose* PKE *is a public-key encryption scheme, and* F_1, F_2 *are circuit-constrained PRFs for* NC^1 *(resp.,* P/poly*). Then, the IPF* $(\mathsf{F}, \mathsf{F}^{-1})$ *from Construction 4.5 is a circuit-constrained IPF for* NC^1 *(resp.,* P/poly*).*

Theorem 4.7. *Suppose* PKE *is a smooth, CCA-secure public-key encryption scheme,* F_1 *is a single-key selective-function-secure circuit-constrained PRF for* NC^1 *(resp.,* P/poly*), and* F_2 *is a single-key, selective-function-secure private circuit-constrained PRF for* NC^1 *(resp.,* P/poly*). Then,* $(\mathsf{F}, \mathsf{F}^{-1})$ *from Construction 4.5 is a single-key, selective-function-secure circuit-constrained IPF for* NC^1 *(resp.,* P/poly*).*

Remark 4.8 (Weaker Constraint Classes). While Construction 4.5 gives a circuit-constrained IPF from private circuit-constrained PRFs, the same construction also applies for building constrained PRFs that support a weaker class of constraints. Specifically, given a private constrained PRF for some constraint family $\mathcal{F}$, if $\mathcal{F}$ is expressive enough to support the decryption operation of a CCA-secure PKE scheme (composed with the constraining function), then the constrained PRF for $\mathcal{F}$ can be leveraged to construct an IPF for the family $\mathcal{F}$ (via Construction 4.5).

Remark 4.9 (Computational Notion of Smoothness). As stated, Theorem 4.7 imposes an additional smoothness requirement (Definition 2.2) on the underlying public-key encryption scheme. While most semantically-secure public-key encryption schemes naturally satisfy this property, a weaker notion of "computational smoothness" also suffices for Theorem 4.7. In particular, we say a public-key encryption scheme $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Encrypt}, \mathsf{PKE.Decrypt})$ with message space $\mathcal{M}$ and ciphertext space $\mathcal{T}$ satisfies computational smoothness if for all messages $m \in \mathcal{M}$ output by an efficient adversary (on input the security parameter λ and the public key pk), and all strings $\mathsf{ct} \in \mathcal{T}$, $\Pr[\mathsf{PKE.Encrypt}(\mathsf{pk}, m) = \mathsf{ct}] = \mathsf{negl}(\lambda)$. Clearly, if PKE is semantically secure, then PKE satisfies computational smoothness. It is straightforward to modify the proof of Theorem 4.7 to rely on the computational version of smoothness. In this case, we can use any CCA-secure public-key encryption scheme to instantiate Construction 4.5.

5 Concrete Instantiations of Constrained IPFs

In this section, we describe how to concretely instantiate Constructions 4.1 and 4.5 using existing lattice-based private constrained PRFs [15,22,24] to

obtain puncturable IPFs and circuit-constrained IPFs (for both NC^1 and $\mathsf{P/poly}$), respectively, from standard lattice assumptions.

Puncturable IPFs from lattices. To apply Construction 4.1, we require an injective puncturable PRF and a private puncturable PRF. As shown in [51], (statistically) injective puncturable PRFs[4] can be built from any one-way function. Next, the recent works of [15,22,24] show how to construct private puncturable PRFs from standard lattice assumptions. Thus, applying Construction 4.1, we obtain puncturable IPFs from standard lattice assumptions. In fact, the construction of Canetti and Chen [24] gives an adaptively-secure private puncturable PRF from the (polynomial) hardness of the learning with errors (LWE) problem [47], and so, combining their construction with Theorem 4.3, we obtain an adaptively-secure puncturable IPF from the (polynomial) hardness of LWE with subexponential error rate.

Circuit-constrained IPFs from lattices. Starting from (single-key) private circuit-constrained PRFs for NC^1 [24] and $\mathsf{P/poly}$ [22], we can leverage Construction 4.5 to obtain (single-key) circuit-constrained IPFs for NC^1 and $\mathsf{P/poly}$, respectively. We give two candidate instantiations based on standard lattice assumptions:

- To construct a circuit-constrained IPF for NC^1-constraints, we require a private circuit-constrained PRF for NC^1 and a CCA-secure public-key encryption scheme with an NC^1 decryption circuit. We can instantiate the private circuit-constrained PRF for NC^1 using the construction of Canetti and Chen [24]. The CCA-secure encryption scheme with NC^1 decryption can be instantiated using existing lattice-based CCA-secure PKE schemes [42,44,45] or by applying the Boneh et al. [14] transformation to a suitable identity-based encryption (IBE) scheme [1,2,25,33] and a message authentication code (MAC) with verification in NC^1, which can be built from lattice-based PRFs [3,4,17]. Putting these pieces together, we obtain a (single-key) circuit-constrained IPF for NC^1 constraints from standard lattice assumptions.
- To construct a circuit-constrained IPF for $\mathsf{P/poly}$, we primarily require a private constrained PRF for $\mathsf{P/poly}$. We instantiate the private circuit-constrained PRF using the recent construction of Brakerski et al. [22], and the CCA-secure public key encryption as above. This yields a secure (single-key) circuit-constrained IPF for general predicates from standard lattice assumptions.

Remark 5.1 (Relaxed Notions of Correctness). Several lattice-based constrained PRF constructions [15,22,23] satisfy a weaker "computational" notion of correctness which roughly states that an efficient adversary with a constrained key k_f cannot find an input x where $\mathsf{F.Eval}(k_f, x) \neq \mathsf{F}(k, x)$, where k is the PRF key. If we instantiate Constructions 4.1 and 4.5 with a constrained PRF that satisfies

[4] A statistically injective puncturable PRF is a puncturable PRF F where $\mathsf{F}(k, \cdot)$ is injective with overwhelming probability over the choice of coins used for sampling the key $k \leftarrow \mathsf{F.Setup}(1^\lambda)$.

a computational notion of correctness, then the resulting constrained IPF also achieves computational correctness. It is straightforward to modify the correctness analysis (Theorems 4.2 and 4.6) to work under a computational notion of correctness. The security analysis remains unchanged since none of the proofs rely on perfect correctness of the underlying constrained PRFs.

6 An Extension: Supporting Delegation

In a delegatable constrained IPF, the holder of a constrained IPF key k_f for a function f can further constrain the key to some function g (i.e., construct a key $k_{f \wedge g}$ that allows IPF evaluation only on points x where $f(x) = g(x) = 1$). Many constrained PRF constructions either support or can be modified to support some flavor of key delegation [19,26,27]. In this section, we describe (informally) how to extend our constrained IPF construction from Sect. 4.2 to support key delegation.

Delegatable constrained PRFs. A constrained PRF that supports one level of delegation can be generically constructed from any constrained PRF by defining the PRF output to be the xor of the outputs of two constrained PRFs. For instance, we can define a PRF F as follows:

$$\mathsf{F}((k_1, k_2), x) := \mathsf{F}_1(k^{(1)}, x) \oplus \mathsf{F}_2(k^{(2)}, x),$$

where F_1 and F_2 are constrained PRFs. The master secret key is $k^{(1)}$ and $k^{(2)}$, and the constrained key for a function f is $(k_f^{(1)}, k^{(2)})$ where $k^{(1)} \leftarrow \mathsf{F}_1.\mathsf{Constrain}(k^{(1)}, f)$. The holder of the constrained key $(k_f^{(1)}, k^{(2)})$ can further constrain to a function of the form $f \wedge g$ by computing $(k_f^{(1)}, k_g^{(2)})$ where $k_g^{(2)} \leftarrow \mathsf{F}_2.\mathsf{Constrain}(k^{(2)}, g)$. Security of this construction follows by a simple hybrid argument. This general technique can be extended to support any a priori polynomially-bounded delegation depth.

Delegatable constrained IPFs. We can define a similar notion of key delegation for constrained IPFs. However, the above method of xoring together the outputs of several constrained IPFs does not directly give a delegatable constrained IPF. In fact, xoring together the outputs of several IPFs may not even give an injective function, let alone an efficiently invertible one. Thus, to support delegation for a constrained IPF, we need a different construction. One method is to use a variant of the xoring trick in conjunction with Construction 4.5. We describe a construction for achieving one level of delegation here. Our construction relies on a CCA-secure public-key encryption scheme PKE, three constrained PRFs F_1, F_2, F_3, and a constrained IPF IPF. The master secret key consists of keys $k^{(1)}$, $k^{(2)}$, $k^{(3)}$ for F_1, F_2, and F_3, respectively, a key $k^{(\mathsf{IPF})}$ for IPF, and the

public/secret key-pair pk, sk for the PKE scheme. Our delegatable IPF works as follows:

$$\mathsf{F}\big((k^{(1)}, k^{(2)}, k^{(3)}, k^{(\mathsf{IPF})}, \mathsf{pk}, \mathsf{sk}),\ x\big) := \left\{ \begin{array}{l} r \leftarrow \mathsf{F}_1(k^{(1)}, x) \oplus \mathsf{F}_3(k^{(3)}, x) \\ \mathsf{ct} \leftarrow \mathsf{PKE.Encrypt}(\mathsf{pk}, x; r) \\ z \leftarrow \mathsf{F}_2(k^{(2)}, \mathsf{ct}) \oplus \mathsf{IPF}(k^{(\mathsf{IPF})}, x) \\ \text{output } (\mathsf{ct}, z) \end{array} \right\}$$

$$\mathsf{F}^{-1}\big((k^{(1)}, k^{(2)}, k^{(3)}, k^{(\mathsf{IPF})}, \mathsf{pk}, \mathsf{sk}),\ (\mathsf{ct}, z)\big) := \left\{ \begin{array}{l} x \leftarrow \mathsf{IPF}^{-1}(k^{(\mathsf{IPF})}, z \oplus \mathsf{F}_2(k^{(2)}, \mathsf{ct})) \\ r \leftarrow \mathsf{F}_1(k^{(1)}, x) \oplus \mathsf{F}_3(k^{(3)}, x) \\ \text{if } \mathsf{ct} \neq \mathsf{PKE.Encrypt}(\mathsf{pk}, x; r) \\ \quad \text{then } x \leftarrow \perp \\ \text{output } x \end{array} \right\}$$

To constrain a key $(k^{(1)}, k^{(2)}, k^{(3)}, k^{(\mathsf{IPF})}, \mathsf{pk}, \mathsf{sk})$ to a function f, we first constrain the PRF keys $k^{(1)}$, $k^{(2)}$ exactly as described in Construction 4.5. In particular, the constrain algorithm computes $k_f^{(1)} \leftarrow \mathsf{F}_1.\mathsf{Constrain}(k^{(1)}, f)$ and $k_F^{(2)} \leftarrow \mathsf{F}_2.\mathsf{Constrain}(k^{(2)}, F_{\mathsf{sk},f})$, where $F_{\mathsf{sk},f}$ is defined as in Eq. (4.1). The constrained key is the tuple $k_f = (k_f^{(1)}, k_F^{(2)}, k^{(3)}, k^{(\mathsf{IPF})}, \mathsf{pk})$. To further constrain (that is, delegate) to a function g, we constrain F_3 and IPF to g. In other words, we compute $k_g^{(3)} \leftarrow \mathsf{F}_3.\mathsf{Constrain}(k^{(3)}, g)$ and $k_g^{(\mathsf{IPF})} \leftarrow \mathsf{IPF.Constrain}(k^{(\mathsf{IPF})}, g)$. The constrained key $k_{f \wedge g}$ for the function $f \wedge g$ is defined to be $k_{f \wedge g} := (k_f^{(1)}, k_F^{(2)}, k_g^{(3)}, k_g^{(\mathsf{IPF})}, \mathsf{pk})$. Security of this construction follows by a similar argument as that used in the proof of Theorem 4.7 (namely, by appealing to security of F_1 and privacy as well as security of F_2), in addition to security of F_3 and the underlying IPF. Our construction can be viewed as taking a standard constrained IPF (that does not support key delegation), and constructing a constrained IPF that supports one level of delegation. Iterating this construction multiple times yields an IPF that can support any a priori bounded number of delegations.

7 Multi-key Constrained IPF from Obfuscation

In this section, we construct a *multi-key* circuit-constrained IPF from (polynomially-hard) indistinguishability obfuscation and one-way functions. Our construction of a circuit-constrained IPF from $i\mathcal{O}$ (and one-way functions) mirrors the Boneh-Zhandry construction [20] of a circuit-constrained PRF from $i\mathcal{O}$ (and one-way functions). More precisely, Boneh and Zhandry show that obfuscating a puncturable PRF effectively gives a circuit-constrained PRF. Similarly, our construction works by obfuscating our punctured IPF construction (Construction 4.1) using $i\mathcal{O}$. In our construction, each constrained IPF key contains two obfuscated programs: one for evaluating the IPF, and one for inverting the IPF. The constraint function f is embedded within the obfuscated evaluation and inversion programs. We now describe our scheme more formally. First, we review the standard definition of indistinguishability obfuscation [5,32].

Definition 7.1 (Indistinguishability Obfuscation [5,32]). *An indistinguishability obfuscator $i\mathcal{O}$ for a circuit class $\mathcal{C}$ is a uniform and efficient algorithm satisfying the following requirements:*

- ***Correctness.*** *For all security parameter $\lambda \in \mathbb{N}$, all circuits $C \in \mathcal{C}$, and all inputs x, we have that*

$$\Pr[C' \leftarrow i\mathcal{O}(C) : C'(x) = C(x)] = 1.$$

– ***Indistinguishability.*** *For all security parameter $\lambda \in \mathbb{N}$, and any two circuits $C_0, C_1 \in \mathcal{C}_\lambda$, if $C_0(x) = C_1(x)$ for all inputs x, then for all efficient adversaries $\mathcal{A}$, we have that*

$$|\Pr[\mathcal{A}(i\mathcal{O}(C_0)) = 1] - \Pr[\mathcal{A}(i\mathcal{O}(C_1)) = 1]| = \mathrm{negl}(\lambda).$$

Construction 7.2. *Fix a domain $\mathcal{X} = \{0,1\}^n$ where $n = n(\lambda)$. Let $\mathsf{F}_1 \colon \mathcal{K}_1 \times \mathcal{X} \to \mathcal{V}$ be a puncturable PRF with key-space $\mathcal{K}_1$ and range $\mathcal{V}$. Let $\mathsf{F}_2 \colon \mathcal{K}_2 \times \mathcal{V} \to \mathcal{X}$ be a puncturable PRF with key-space $\mathcal{K}_2$. The constrained IPF $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ with key-space $\mathcal{K} = \mathcal{K}_1 \times \mathcal{K}_2$, domain $\mathcal{X}$, and range $\mathcal{Y} = \mathcal{V} \times \mathcal{X}$ is defined as follows:*

– *The IPF key is a pair of keys $k = (k^{(1)}, k^{(2)}) \in \mathcal{K}_1 \times \mathcal{K}_2 = \mathcal{K}$. On input a key $(k^{(1)}, k^{(2)})$ and an input $x \in \mathcal{X}$, the value of the IPF is defined to be*

$$\mathsf{F}(k, x) := \Big(\mathsf{F}_1(k^{(1)}, x),\ \ \mathsf{F}_2(k^{(2)}, \mathsf{F}_1(k^{(1)}, x)) \oplus x\Big).$$

– *On input $k = (k^{(1)}, k^{(2)}) \in \mathcal{K}_1 \times \mathcal{K}_2 = \mathcal{K}$, and $y = (y_1, y_2) \in \mathcal{V} \times \mathcal{X} = \mathcal{Y}$, the inversion algorithm $\mathsf{F}^{-1}(k, y)$ first computes $x \leftarrow \mathsf{F}_2(k^{(2)}, y_1) \oplus y_2$ and outputs*

$$\mathsf{F}^{-1}(k, (y_1, y_2)) := \begin{cases} x & \text{if } y_1 = \mathsf{F}_1(k^{(1)}, x) \\ \perp & \text{otherwise.} \end{cases}$$

Next, we define the setup and constraining algorithms for the IPF $(\mathsf{F}, \mathsf{F}^{-1})$.

– $\mathsf{F}.\mathsf{Setup}(1^\lambda)$*: On input the security parameter λ, the setup algorithm samples two puncturable PRF keys $k^{(1)} \leftarrow \mathsf{F}_1.\mathsf{Setup}(1^\lambda)$ and $k^{(2)} \leftarrow \mathsf{F}_2.\mathsf{Setup}(1^\lambda)$, and outputs $k = (k^{(1)}, k^{(2)})$.*
– $\mathsf{F}.\mathsf{Constrain}(k, f)$*: On input the IPF key $k = (k^{(1)}, k^{(2)})$ and a constraint function $f \in \mathcal{F}$, the constrain algorithm outputs two obfuscated programs $P_0 = i\mathcal{O}(P^{\mathsf{Eval}}[f, k^{(1)}, k^{(2)}])$ and $P_1 = i\mathcal{O}(P^{\mathsf{Inv}}[f, k^{(1)}, k^{(2)}])$ where the programs $P^{\mathsf{Eval}}[f, k^{(1)}, k^{(2)}]$ and $P^{\mathsf{Inv}}[f, k^{(1)}, k^{(2)}]$ are defined in Figs. 1 and 2. Note that the programs P^{Eval} and P^{Inv} are padded to the maximum size of any program that appears in the proof of Theorem 7.4.*
– $\mathsf{F}.\mathsf{Eval}(k_f, x)$*: On input the constrained key $k_f = (P_1, P_2)$, and a point $x \in \mathcal{X}$, the evaluation algorithm outputs $P_1(x)$.*
– $\mathsf{F}.\mathsf{Eval}^{-1}(k_f, y)$*: On input the constrained key $k_f = (P_1, P_2)$, and a point $y \in \mathcal{Y}$, the inversion algorithm outputs $P_2(y)$.*

We now state our correctness and security theorems. We provide the formal proofs in the full version [16].

Theorem 7.3. *Suppose F_1 and F_2 are puncturable PRFs, and $i\mathcal{O}$ is an indistinguishability obfuscator. Then, the IPF $(\mathsf{F}, \mathsf{F}^{-1})$ from Construction 7.2 is correct.*

Theorem 7.4. *Suppose F_1 and F_2 are selectively-secure puncturable PRFs, $i\mathcal{O}$ is an indistinguishability obfuscator, and $|\mathcal{X}| / |\mathcal{V}| = \mathrm{negl}(\lambda)$. Then $(\mathsf{F}, \mathsf{F}^{-1})$ from Construction 7.2 is a selectively-secure circuit-constrained IPF.*

Constants: *a function* $f \in \mathcal{F}$*, and two keys* $k^{(1)}$ *and* $k^{(2)}$ *for* F_1 *and* F_2*, respectively.*

On input $x \in \mathcal{X}$*:*

1. *If* $f(x) = 0$*, output* $\perp$*.*
2. *Otherwise, output* $\mathsf{F}(k, x) = \big(\mathsf{F}_1(k^{(1)}, x),\ \mathsf{F}_2(k^{(2)}, \mathsf{F}_1(k^{(1)}, x)) \oplus x\big)$*.*

Fig. 1. The program $P^{\mathsf{Eval}}[f, k^{(1)}, k^{(2)}]$

Constants: *a function* $f \in \mathcal{F}$*, and two keys* $k^{(1)}$ *and* $k^{(2)}$ *for* F_1 *and* F_2*, respectively.*

On input $y = (y_1, y_2) \in \mathcal{V} \times \mathcal{X}$

1. *Compute* $x \leftarrow \mathsf{F}_2(k^{(2)}, y_1) \oplus y_2$*.*
2. *If* $f(x) = 0$ *or* $y_1 \neq \mathsf{F}_1(k^{(1)}, x)$*, output* $\perp$*.*
3. *Otherwise, output* x*.*

Fig. 2. The program $P^{\mathsf{Inv}}[f, k^{(1)}, k^{(2)}]$

Acknowledgments. We thank the anonymous TCC reviewers for helpful comments on this work. This work was funded by NSF, the DARPA/ARL SAFEWARE project, a grant from ONR, and the Simons Foundation. Opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_28
2. Agrawal, S., Boneh, D., Boyen, X.: Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 98–115. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_6
3. Banerjee, A., Peikert, C.: New and improved key-homomorphic pseudorandom functions. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 353–370. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_20

4. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_42
5. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1
6. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_30
7. Bellare, M., Fischlin, M., O'Neill, A., Ristenpart, T.: Deterministic encryption: definitional equivalences and constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 360–378. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_20
8. Bellare, M., Fuchsbauer, G.: Policy-based signatures. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 520–537. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_30
9. Bellare, M., Hofheinz, D., Kiltz, E.: Subtleties in the definition of IND-CCA: when and how should challenge decryption be disallowed? J. Cryptol. **28**(1), 29–48 (2015)
10. Bellare, M., Namprempre, C.: Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44448-3_41
11. Bellare, M., Rogaway, P.: Encode-then-encipher encryption: how to exploit nonces or redundancy in plaintexts for efficient cryptography. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 317–330. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44448-3_24
12. Boldyreva, A., Fehr, S., O'Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_19
13. 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). https://doi.org/10.1007/978-3-540-24676-3_14
14. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. SIAM J. Comput. **36**(5), 1301–1328 (2007)
15. Boneh, D., Kim, S., Montgomery, H.: Private puncturable PRFs from standard lattice assumptions. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 415–445. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_15
16. Boneh, D., Kim, S., Wu, D.J.: Constrained keys for invertible pseudorandom functions. Cryptology ePrint Archive, Report 2017/477 (2017). http://eprint.iacr.org/2017/477
17. Boneh, D., Lewi, K., Montgomery, H.W., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 410–428. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_23

18. Boneh, D., Lewi, K., Wu, D.J.: Constraining pseudorandom functions privately. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10175, pp. 494–524. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54388-7_17
19. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_15
20. Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 480–499. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_27
21. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_29
22. Brakerski, Z., Tsabary, R., Vaikuntanathan, V., Wee, H.: Private constrained PRFs (and more) from LWE. In: TCC (2017)
23. Brakerski, Z., Vaikuntanathan, V.: Constrained key-homomorphic PRFs from standard lattice assumptions - or: how to secretly embed a circuit in your PRF. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 1–30. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_1
24. Canetti, R., Chen, Y.: Constraint-hiding constrained PRFs for NC^1 from LWE. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 446–476. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_16
25. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_27
26. Chandran, N., Raghuraman, S., Vinayagamurthy, D.: Constrained pseudorandom functions: verifiable and delegatable. IACR Cryptology ePrint Archive 2014 (2014)
27. Datta, P., Dutta, R., Mukhopadhyay, S.: Constrained pseudorandom functions for unconstrained inputs revisited: achieving verifiability and key delegation. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10175, pp. 463–493. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54388-7_16
28. Deshpande, A., Koppula, V., Waters, B.: Constrained pseudorandom functions for unconstrained inputs. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 124–153. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_5
29. Fuchsbauer, G.: Constrained verifiable random functions. IACR Cryptology ePrint Archive 2014 (2014)
30. Fuchsbauer, G., Konstantinov, M., Pietrzak, K., Rao, V.: Adaptive security of constrained PRFs. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 82–101. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_5
31. Fuller, B., O'Neill, A., Reyzin, L.: A unified approach to deterministic encryption: new constructions and a connection to computational entropy. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 582–599. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_33
32. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013)
33. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC (2008)

34. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. In: FOCS (1984)
35. Hofheinz, D.: Fully secure constrained pseudorandom functions using random oracles. IACR Cryptology ePrint Archive 2014 (2014)
36. Hohenberger, S., Koppula, V., Waters, B.: Adaptively secure puncturable pseudorandom functions in the standard model. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 79–102. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_4
37. Iwata, T., Yasuda, K.: BTM: a single-key, inverse-cipher-free mode for deterministic authenticated encryption. In: Jacobson, M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 313–330. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-05445-7_20
38. Iwata, T., Yasuda, K.: HBS: a single-key mode of operation for deterministic authenticated encryption. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 394–415. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03317-9_24
39. Katz, J., Yung, M.: Unforgeable encryption and chosen ciphertext secure modes of operation. In: Goos, G., Hartmanis, J., van Leeuwen, J., Schneier, B. (eds.) FSE 2000. LNCS, vol. 1978, pp. 284–299. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44706-7_20
40. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: ACM CCS (2013)
41. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM J. Comput. **17**(2), 373–386 (1988)
42. Micciancio, D., Peikert, C.: Trapdoors for lattices: simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_41
43. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC (1990)
44. Peikert, C.: Bonsai trees (or, arboriculture in lattice-based cryptography). IACR Cryptology ePrint Archive 2009 (2009)
45. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: STOC (2008)
46. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_35
47. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC (2005)
48. Rogaway, P.: Authenticated-encryption with associated-data. In: ACM CCS (2002)
49. Rogaway, P., Bellare, M., Black, J.: OCB: a block-cipher mode of operation for efficient authenticated encryption. ACM Trans. Inf. Syst. Secur. (TISSEC) **6**(3), 365–403 (2003)
50. Rogaway, P., Shrimpton, T.: Deterministic authenticated encryption: a provable-security treatment of the key-wrap problem. In: EUROCRYPT (2006)
51. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: STOC (2014)