This is the full version of the extended abstract that appears in the proceedings of CT-RSA 2016.

Constrained PRFs for Unbounded Inputs

Hamza Abusalah* Georg Fuchsbauer* Krzysztof Pietrzak*

Institute of Science and Technology Austria
{habusalah,gfuchsbauer,pietrzak}@ist.ac.at

Abstract

A constrained pseudorandom function $F\colon \mathcal{K}\times\mathcal{X}\to\mathcal{Y}$ for a family $\mathcal{T}\subseteq 2^{\mathcal{X}}$ of subsets of $\mathcal{X}$ is a function where for any key $k\in\mathcal{K}$ and set $S\in\mathcal{T}$ one can efficiently compute a constrained key k_S which allows to evaluate $F(k,\cdot)$ on all inputs $x\in S$, while even given this key, the outputs on all inputs $x\notin S$ look random. At Asiacrypt'13 Boneh and Waters gave a construction which supports the most general set family so far. Its keys k_C are defined for sets decided by boolean circuits C and enable evaluation of the PRF on any $x\in\mathcal{X}$ where $C(x)=1$. In their construction the PRF input length and the size of the circuits C for which constrained keys can be computed must be fixed beforehand during key generation.

We construct a constrained PRF that has an unbounded input length and whose constrained keys can be defined for any set recognized by a Turing machine. The only a priori bound we make is on the description size of the machines. We prove our construction secure assuming public-coin differing-input obfuscation.

As applications of our constrained PRF we build a broadcast encryption scheme where the number of potential receivers need not be fixed at setup (in particular, the length of the keys is independent of the number of parties) and the first identity-based non-interactive key exchange protocol with no bound on the number of parties that can agree on a shared key.

Keywords: Constrained PRFs, broadcast encryption, identity-based non-interactive key exchange.

1 Introduction

Constrained PRFs. A pseudorandom function (PRF) [GGM86] is a keyed function $\mathsf{F}\colon \mathcal{K}\times\mathcal{X}\to\mathcal{Y}$ for which no efficient adversary, given access to an oracle $\mathcal{O}(\cdot)$, can distinguish the case where $\mathcal{O}(\cdot)$ is $\mathsf{F}(k,\cdot)$ with a random key $k\in\mathcal{K}$ from the case where $\mathcal{O}(\cdot)$ is a uniformly random function $\mathcal{X}\to\mathcal{Y}$.

Three papers [BW13, BGI14, KPTZ13] independently introduce the concept of a *constrained* PRF. Consider a set $\mathcal{P}$, where each $v\in\mathcal{P}$ specifies some predicate $p_v\colon \mathcal{X}\to\{0,1\}$ that defines a (potentially exponential-size) subset $S_v=\{x\in\mathcal{X}\mid p_v(x)=1\}$. A constrained PRF for a predicate family $\mathcal{P}$ is a PRF F with an additional constrain algorithm $k_v\leftarrow\mathsf{F}.\mathsf{Constr}(k,v)$ that on input a key $k\in\mathcal{K}$ and a predicate $v\in\mathcal{P}$ outputs a constrained key k_v that can be used to compute $\mathsf{F}(k,x)$ on all $x\in S_v$, while, given this key, all values $\mathsf{F}(k,x)$ for $x\notin S_v$ still look random.

Constrained PRFs (CPRF) have been constructed for several interesting predicates. All three papers [BW13, BGI14, KPTZ13] show that the classical GGM construction [GGM86] of a PRF with input domain $\{0,1\}^n$ yields a *prefix*-constrained PRF. This means $\mathcal{P}=\{0,1\}^{\le n}$ and for any $v\in\mathcal{P}$ the derived key k_v allows to compute $\mathsf{F}(k,x)$ for all x with prefix v, i.e., $x=v\|x'\in\{0,1\}^n$ for some x'. Assuming (leveled) multilinear maps [GGH13a, CLT13, LSS14], Boneh and Waters [BW13] construct CPRFs for much more general set systems. They present a bit-fixing PRF, where $\mathcal{P}=\{0,1,?\}^n$ and for $v\in\mathcal{P}$ we have $p_v(x)=1$ if x agrees with v on all indices different from '?',

*Supported by the European Research Council, ERC Starting Grant (259668-PSPC)

i.e., for all $i = 1, \ldots, n$, either $v[i] = ?$ or $v[i] = x[i]$. They moreover construct a circuit-constrained PRF, where the predicates are arbitrary circuits $C\colon \{0,1\}^n \to \{0,1\}$ of some fixed depth.

CPRFs have already found many interesting applications. From a prefix CPRF, one can construct a puncturable PRF [SW14], which is a constrained PRF for predicates $\mathcal{P} = \{0,1\}^n$ where for $v \in \mathcal{P}$, the key k_v lets one compute $\mathsf{F}(k,x)$ on all $x \neq v$. The GGM construction yields a puncturable PRF with punctured keys whose length is linear in the PRF input length. Puncturable PRFs play a crucial role in the security proofs of most of the recent constructions based on indistinguishability obfuscation [BGI+12, GGH+13b], and we will also use them in this paper.

The more general bit-fixing and circuit-constrained PRFs can be used to construct a variety of sophisticated cryptographic tools including broadcast encryption (BE) and identity-based non-interactive key-exchange, as outlined next.

Broadcast encryption. In a BE scheme [FN94, YFDL04, BGW05, BH08, PPS11, BWZ14] there is a set of n users, and for any given subset $S \subseteq \{1, \ldots, n\}$ of users, we want to be able to encrypt a message (as a short ciphertext) that can be decrypted only by the users included in S. This can be achieved using a bit-fixing PRF with domain $\{0,1\}^n$: Sample a random key k, and give a constrained key k_{v_i} to user i where $v_i[i] = 1$ and $v_i[j] = ?$ for any $j \neq i$. Thus, k_{v_i} allows to evaluate the PRF on exactly those inputs with a '1' in position i.

To broadcast a message m to a set S of users, we simply send a symmetric encryption of m under the key $\mathsf{F}(k, x_S)$, where $x_S[i] = 1$ if $i \in S$ and $x_S[i] = 0$ otherwise. Note that user i can compute $\mathsf{F}(k, x_S)$ (and thus decrypt) iff her key k_{v_i} satisfies $v_i[i] = x_S[i]$, which by construction holds iff $i \in S$.

Non-interactive key exchange. In an identity-based non-interactive key exchange (ID-NIKE) [SOK00, FHPS13, BW13] scheme there are parties that each have some identity $id \in \{0,1\}^\ell$. For any set S of at most n parties we want the parties in S to be able to locally compute a shared key K_S which is indistinguishable from random for all parties outside of S. Such a scheme can be constructed from a bit-fixing PRF F with domain $\{0,1\}^{n\cdot\ell}$. At setup, sample a key k for F and give to party $id \in \{0,1\}^\ell$ a set of n constrained keys $k_{id}^{(1)}, \ldots, k_{id}^{(n)}$, where $k_{id}^{(i)}$ is a key for the set $?^{(i-1)\ell} \| id \| ?^{(n-i)\ell}$. Now, only parties $id_1, \ldots, id_n$ can compute the joint key $K_S := \mathsf{F}(k, id_1 \| id_2 \| \ldots \| id_n)$.

CPRFs with unbounded input length. The disadvantage of the BE and ID-NIKE constructions just outlined is that the number n of possible recipients (for BE) or parties agreeing on a key (for ID-NIKE) must be fixed when setting up the system. Moreover, the length of the constrained keys given to every user is at least linear in n.

In this paper we construct a constrained PRF for which there is no a priori bound on the input length. The constraints on keys are given by Turing machines (TM), that is, given a key k and a machine M, we can derive a constrained key k_M that allows to compute $\mathsf{F}(k,x)$ for any input x for which $M(x) = 1$. The only thing that must be bounded beforehand is the *size* of the TMs that we want to support. In our construction a constrained key for a machine M will be an obfuscated circuit whose size only depends on the size of M.

Adaptive vs. selective security. Security of constrained PRFs is defined via a game in which a challenger picks a random key k and the adversary chooses $x^* \in \mathcal{X}$ and must distinguish $\mathsf{F}(k, x^*)$ from random. The adversary has also access to oracles to query constrained keys for sets $S \not\ni x^*$. We prove *selective* security of our CPRF where the adversary must choose x^* before it can query constrained keys. From a selectively secure CPRF we can get an *adaptively* secure CPRF (where the adversary can decide on x^* after its key queries) via "complexity leveraging"—but this reduction loses a factor that is exponential in the input length. Proving adaptive security for CPRFs without an exponential security loss is generally hard and Fuchsbauer et al. [FKPR14]

show that for the bit-fixing CPRF from [BW13] any "simple" security reduction must lose an exponential factor.

Adaptive security of CPRFs was proved for the GGM-based prefix-constrained PRF in [FKPR14] losing only a quasi-polynomial (rather than an exponential) factor. Moreover, Hohenberger, Koppula and Waters [HKW14] construct an adaptively secure puncturable PRF with polynomial security loss using indistinguishability obfuscation ($i\mathcal{O}$) [GGH+13b, SW14, PST14]. Hofheinz et al. [HKKW14] construct an adaptively secure bit-fixing PRF, also using $i\mathcal{O}$, and additionally relying on the random-oracle model. We leave the construction of adaptively secure constrained unbounded-length PRFs (for any interesting set of constraints) as a challenging open problem.

Applications. As two applications of our constrained PRFs we show that they directly yield broadcast encryption and ID-NIKE for an unbounded number of parties. In particular, all parameters (private/public key size and for BE also ciphertext overhead) are poly-logarithmic in the number of potential parties (or equivalently, polynomial in the length of the identities). For BE, this has only recently been achieved by Boneh, Waters and Zhandry [BWZ14], who construct a BE scheme supporting n parties directly from $O(\log(n))$-way multilinear maps. For ID-NIKE, our construction is the first to achieve this; all previous schemes require the maximum size of the group of users agreeing on a key to be fixed at setup, and they have parameters that depend at least linearly on this size.

A circuit-constrained PRF. A first idea for constructing a constrained PRF is to start with a standard PRF F; given a key k and a set S, we can define a constrained key as a program P_S which on input x checks whether $x \in S$, and if so, outputs $\mathsf{F}(k, x)$. We cannot define the constrained key as the program P_S as such, since an adversary could extract the key k from P_S, and hence $\mathsf{F}(k, \cdot)$ would not be pseudorandom for $x \notin S$ given P_S.

When S is decided by a circuit, the above issue can be avoided by *obfuscating* P_S before outputting it. The strong notion of *virtual black-box obfuscation*, which requires that an obfuscated program leaks nothing about the program apart from its input/output behavior, is not achievable for general functionalities [BGI+12]. We therefore use *indistinguishability obfuscation* ($i\mathcal{O}$), which only guarantees that obfuscations of two circuits (of the same size) that output the same on any input are indistinguishable. A candidate $i\mathcal{O}$ scheme was proposed by Garg et al. [GGH+13b]. Although the notion seems weak, it has proven to be surprisingly useful.

Consider a CPRF derived from a puncturable PRF F for which a constrained key k_C for a circuit C is defined as an $i\mathcal{O}$ obfuscation of the circuit P_C, which on input x returns $\mathsf{F}(k, x)$ if $C(x) = 1$ and $\perp$ otherwise. In the selective-security game an adversary $\mathcal{A}$ chooses an input x^*, can then ask for constrained keys for circuits C with $C(x^*) = 0$ and must distinguish $\mathsf{F}(k, x^*)$ from random. In the security proof we first define a modified game where $\mathcal{A}$, when asking for a constrained key for a circuit C, is given an $i\mathcal{O}$ obfuscation of a circuit P'_C that outputs $\mathsf{F}(k_{x^*}, x)$ if $C(x) = 1$ and $\perp$ otherwise. The difference between P_C and P'_C is that in the latter F is evaluated using a key k_{x^*} that is punctured at x^*.

Recall that the adversary can only submit circuits C with $C(x^*) = 0$ to its oracle. For every such C we have $P_C(x^*) = P'_C(x^*) = \perp$, and on any other input x, P_C and P'_C also return the same output (namely $\mathsf{F}(k, x)$ if $C(x) = 1$ and $\perp$ otherwise). By security of $i\mathcal{O}$, obfuscations of P_C and P'_C are thus indistinguishable, which means that the modified game is indistinguishable from the original game. From an adversary $\mathcal{A}$ winning the modified game we obtain an adversary $\mathcal{B}$ that breaks the puncturable PRF F: When $\mathcal{A}$ commits to x^*, $\mathcal{B}$ asks for a punctured key k_{x^*}, which allows $\mathcal{B}$ to answer $\mathcal{A}$'s constrained-key queries in the modified game. If $\mathcal{A}$ distinguishes $\mathsf{F}(k, x^*)$ from random then so does $\mathcal{B}$.

A TM-constrained PRF. The above construction uses $i\mathcal{O}$ for circuits. Recently, Koppula, Lewko and Waters [KLW15] constructed $i\mathcal{O}$ for Turing machines. However, we cannot simply

replace circuits by TMs in the construction just sketched. In the security proof we need to upper-bound the size of the TM to be obfuscated when we switch from a TM using k to a TM using k_{x^*}. This is however impossible because the size of the underlying punctured key k_{x^*} cannot be a priori bounded for unbounded inputs x^*.

To overcome this problem, we use a collision-resistant hash function H to map long inputs to inputs of fixed length. Concretely, we define our CPRF as $\mathsf{F}(k, x) := \mathsf{PF}(k, H(x))$, where PF is a puncturable PRF. Consequently, a constrained key would be an obfuscation of the TM P_M that checks the input legitimacy of x, i.e., whether $M(x) = 1$, and evaluates PF on $H(x)$. In order to give a reduction of security to the puncturable PRF PF, we would, as before, replace the obfuscation of P_M by one of P'_M, which uses $k_{H(x^*)}$ instead of k. While this solves the size-bounding problem, it poses new challenges. Namely, $i\mathcal{O}$ is not sufficient as P_M and P'_M are in general not functionally equivalent: consider a machine M with $M(x^*) = 0$ and $M(x) = 1$ for some x with $H(x) = H(x^*)$; then $P_M(x) = \mathsf{F}(k, x)$, whereas $P'_M(x) = \bot$.

Differing-input obfuscation. Instead of $i\mathcal{O}$, we resort to a stronger form of obfuscation. Whereas $i\mathcal{O}$ yields indistinguishable obfuscations of programs that are functionally equivalent, differing-input obfuscation ($di\mathcal{O}$, a.k.a. extractability obfuscation) introduced by [BGI+12, BST14] for circuits and later by [BCP14, ABG+13] for TMs, guarantees that from an adversary that distinguishes obfuscations of two circuits (or TMs), one can extract an input on which they differ. $di\mathcal{O}$ is a strong assumption and in fact was shown to be implausible to exist [GGHW14]. We will use a weaker assumption suggested by Ishai, Pandey and Sahai [IPS15] and called public-coin $di\mathcal{O}$, for which no such implausibility results are known. Informally, this notion only implies indistinguishability for pairs of programs if it is hard to find an input on which the two programs differ *even when given the randomness used to sample this pair of circuits.*

We replace $i\mathcal{O}$ in our CPRF construction by public-coin $di\mathcal{O}$ for TMs with unbounded inputs from [IPS15] and define a constrained key for M as a $di\mathcal{O}$ obfuscation of P_M. This solution is not without limitations; a constrained key is now a $di\mathcal{O}$-obfuscated TM and therefore keys are large and their size depends on the running time of the constraining TM M, which we show how to avoid next.

SNARKs. We "outsource" the check of input legitimacy (whether x satisfies $M(x) = 1$) to the party that evaluates the PRF. The latter first computes a succinct non-interactive argument of knowledge (SNARK) of a legitimate x and passes this SNARK to the obfuscated program. A SNARK system is a computationally sound non-interactive proof of knowledge for which proofs are universally succinct. That is, the length of a proof π for a statement η as well as its verification time are bounded by an a-priori-fixed polynomial in the length $|\eta|$ of the statement. In particular, we use a SNARK system for the language $L_{legit} := \{(H, M, h) \mid \exists\, x : M(x) = 1 \wedge H(x) = h\}$.

Instead of running M, the program P_M now only needs to verify a SNARK, which can be implemented by a *circuit*; we thus only require obfuscation of circuits. Let P_M be the circuit that on input (h, π) outputs $\mathsf{PF}(k, h)$ iff π is a valid SNARK for (H, M, h). A constrained key is then a public-coin $di\mathcal{O}$ obfuscation of P_M, whose size only depends on the size of M but not on its running time.

As we use public-coin $di\mathcal{O}$, we require the hash function H to be public-coin [HR04], that is, collision-resistance (CR) must hold when the adversary is given the randomness used to sample a hash function from the family; moreover, the SNARK must be in the common random string model. Such hash functions and SNARKs exist as discussed in [IPS15]. Assuming puncturable PRFs, public-coin CR hash functions, a SNARK for the language L_{legit} and public-coin $di\mathcal{O}$ for circuits, our construction is a TM-constrained PRF for inputs of unbounded length.

$\underline{\mathbf{Exp}^{\mathcal{O},b}_{\mathcal{F},\mathcal{A}}(\lambda)}$
$k \leftarrow \mathsf{F.Smp}(1^\lambda)$; $C, E := \emptyset$
$(x^*, st) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(1^\lambda)$
If $x^* \in E$ then abort
If $b = 1$ then $y := \mathsf{F.Eval}(k, x^*)$
Else $y \leftarrow \mathcal{Y}$
$C := C \cup \{x^*\}$
Return $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(st, y)$

Oracle $\textsc{Constr}(S)$
If $S \notin \mathcal{S}_\lambda \vee S \cap C \neq \emptyset$
  Return $\bot$
$E := E \cup S$
$k_S \leftarrow \mathsf{F.Constr}(k, S)$
Return k_S

Oracle $\textsc{Eval}(x)$
If $x \notin \mathcal{X} \vee x \in C$
  Return $\bot$
$E := E \cup \{x\}$
$y = \mathsf{F.Eval}(k, x)$
Return y

Figure 1: The security game for constrained PRFs.

2 Preliminaries

2.1 Constrained and Puncturable PRFs

Definition 1 (Constrained Functions)**.** *A family of keyed functions $\mathcal{F}_\lambda = \{\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}\}$ over a key space $\mathcal{K}$, a domain $\mathcal{X}$ and a range $\mathcal{Y}$ is* efficiently computable *if there exist a PPT sampler* $\mathsf{F.Smp}$ *and a deterministic polynomial-time (PT) evaluator* $\mathsf{F.Eval}$ *such that:*

- $k \leftarrow \mathsf{F.Smp}(1^\lambda)$*: On input a security parameter λ,* $\mathsf{F.Smp}$ *outputs a key $k \in \mathcal{K}$.*
- $\mathsf{F}(k, x) := \mathsf{F.Eval}(k, x)$*, for $k \in \mathcal{K}$ and $x \in \mathcal{X}$.*

We say $\mathcal{F}_\lambda$ is constrained *w.r.t. a family $\mathcal{S}_\lambda$ of subsets of $\mathcal{X}$, with constrained key space $\mathcal{K}_\mathcal{S}$ such that $\mathcal{K}_\mathcal{S} \cap \mathcal{K} = \emptyset$, if* $\mathsf{F.Eval}$ *accepts inputs from $(\mathcal{K} \cup \mathcal{K}_\mathcal{S}) \times \mathcal{X}$ and there exists the following PPT algorithm:*

- $k_S \leftarrow \mathsf{F.Constr}(k, S)$*: On input a key $k \in \mathcal{K}$ and a description*[1] *of a set $S \in \mathcal{S}_\lambda$,* $\mathsf{F.Constr}$ *outputs a constrained key $k_S \in \mathcal{K}_\mathcal{S}$ such that*

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

Definition 2 (Security of Constrained PRFs)**.** *A family of (efficiently computable) constrained functions $\mathcal{F}_\lambda = \{\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}\}$ is selectively* pseudorandom*, if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in $\mathbf{Exp}^{\mathcal{O},b}_{\mathcal{F},\mathcal{A}}$, defined in Fig. 1, with $\mathcal{O}_1 = \emptyset$ and $\mathcal{O}_2 = \{\textsc{Constr}(\cdot), \textsc{Eval}(\cdot)\}$, it holds that*

$$\mathsf{Adv}^{\mathcal{O}}_{\mathcal{F},\mathcal{A}}(\lambda) := \left| \Pr\left[\mathbf{Exp}^{\mathcal{O},0}_{\mathcal{F},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathcal{O},1}_{\mathcal{F},\mathcal{A}}(\lambda) = 1\right] \right| = \mathsf{negl}(\lambda) \ . \tag{1}$$

$\mathcal{F}_\lambda$ is adaptively pseudorandom *if the same holds for $\mathcal{O}_1 = \mathcal{O}_2 = \{\textsc{Constr}(\cdot), \textsc{Eval}(\cdot)\}$.*

Puncturable PRFs [SW14] are a simple type of constrained PRFs whose domain is $\{0,1\}^n$ for some n, and constrained keys can only be derived for the sets $\{\{0,1\}^n \setminus \{x_1, \ldots, x_m\} \mid x_1, \ldots, x_m \in \{0,1\}^n, m = \mathsf{poly}(\lambda)\}$, i.e., a punctured key $k_{x_1 \ldots x_m}$ can evaluate the PRF on all inputs *except* on polynomially many $x_1, \ldots, x_m$. We only require pseudorandomness to hold against selective adversaries.

Definition 3 (Puncturable PRFs [SW14])**.** *A family of PRFs $\mathcal{F}_\lambda = \{\mathsf{F}\colon \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}\}$ is* puncturable *if it is constrainable for sets $\{0,1\}^n \setminus T$, where $T \subseteq \{0,1\}^n$ is of polynomial size. $\mathcal{F}_\lambda$ is (selectively) pseudorandom if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in $\mathbf{Exp}^{\text{PCT-}b}_{\mathcal{F},\mathcal{A}}(\lambda)$, defined in Fig. 2, we have*

$$\mathsf{Adv}^{\text{PCT}}_{\mathcal{F},\mathcal{A}}(\lambda) := \left| \Pr\left[\mathbf{Exp}^{\text{PCT-0}}_{\mathcal{F},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{PCT-1}}_{\mathcal{F},\mathcal{A}}(\lambda) = 1\right] \right| = \mathsf{negl}(\lambda) \ .$$

[1] As outlined in the introduction, we assume that every set in $\mathcal{S}$ can be specified by a short and efficiently computable predicate.

$\underline{\mathbf{Exp}^{\text{PCT-}b}_{\mathcal{F},\mathcal{A}}(\lambda)}$

$(x^*, T, st) \leftarrow \mathcal{A}_1(1^\lambda)$; If $x^* \notin T$, then abort
$k \leftarrow \mathsf{F.Smp}(1^\lambda)$; $k_{\overline{T}} \leftarrow \mathsf{F.Constr}(k, \{0,1\}^n \setminus T)$
If $b = 1$ then $y := \mathsf{F.Eval}(k, x^*)$; else $y \leftarrow \mathcal{Y}$
Return $b' \leftarrow \mathcal{A}_2^{\text{EVAL}(\cdot)}(st, k_{\overline{T}}, y)$

$\underline{\textbf{Oracle } \text{EVAL}(x)}$

If $x = x^*$, return $\perp$
Return $\mathsf{F.Eval}(k, x)$

Figure 2: The selective-security game for puncturable PRFs.

Puncturable PRFs are easily obtained from prefix-constrained PRFs, which were constructed from the GGM PRF [GGM86] in [BW13, BGI14, KPTZ13].

2.2 Collision-Resistant Hash Functions

A family of hash functions is collision-resistant (CR) if given a uniformly sampled function, it is hard to find inputs on which it collides. It is called public-coin CR if this is hard even when given the coins used to sample the function.

Definition 4 (Public-Coin CR Hash Functions [HR04])**.** *A family of (efficiently computable) functions $\mathcal{H}_\lambda = \{H\colon \{0,1\}^* \to \{0,1\}^n\}$, for which Smp samples a random function, is public-coin collision-resistant if for every PPT adversary $\mathcal{A}$ it holds that*

$$\Pr\left[\begin{array}{l} r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; H := \mathsf{Smp}(1^\lambda, r); \\ (x_1, x_2) \leftarrow \mathcal{A}(1^\lambda, r) \end{array} : \begin{array}{r} H(x_1) = H(x_2) \\ \wedge\ x_1 \neq x_2 \end{array}\right] = \mathsf{negl}(\lambda) \ .$$

2.3 Indistinguishability and Differing-Input Obfuscation

As a consequence of the impossibility of virtual black-box obfuscation, Barak et al. [BGI+12], proposed two weaker notions: *indistinguishability obfuscation* ($i\mathcal{O}$) and *differing-input obfuscation* ($di\mathcal{O}$). The first, $i\mathcal{O}$, guarantees that obfuscations of equivalent functionalities are computationally indistinguishable.

Definition 5 (iO [GGH+13b])**.** *A uniform PPT algorithm iO is an indistinguishability obfuscator for a family of polynomial-size circuits $\mathcal{C}_\lambda$, if the following hold:*

- *For all $\lambda \in \mathbb{N}$, $C \in \mathcal{C}_\lambda$, and x:* $\Pr\left[\widetilde{C} \leftarrow \mathsf{iO}(1^\lambda, C) :\ C(x) = \widetilde{C}(x)\right] = 1$.
- *For every PPT adversary $\mathcal{A}$ and all $C_0, C_1 \in \mathcal{C}_\lambda$ s.t. $\forall x$, $C_0(x) = C_1(x)$:*

$$\left|\Pr\left[\mathcal{A}\big(\mathsf{iO}(1^\lambda, C_0)\big) = 1\right] - \Pr\left[\mathcal{A}\big(\mathsf{iO}(1^\lambda, C_1)\big) = 1\right]\right| = \mathsf{negl}(\lambda) \ .$$

Differing-input obfuscation is a stronger notion which requires that for any efficient adversary that distinguishes obfuscations of two functionalities, there exists an efficient *extractor* $\mathcal{E}$ that extracts a point on which the functionalities differ. Ishai, Pandey and Sahai [IPS15] weaken this to *public-coin* $di\mathcal{O}$, where the extractor is given the coins used to sample the functionalities. We will use this concept for circuits, which is formalized by requiring that indistinguishability only holds for circuits output by a sampler Samp for which no differing-input extractor exists.

Definition 6 (Public-Coin Differing-Input Sampler [IPS15])**.** *A non-uniform PPT sampler Samp is a public-coin differing-input sampler for a polynomial-size family of circuits $\mathcal{C}_\lambda$ if the output of Samp is in $\mathcal{C}_\lambda \times \mathcal{C}_\lambda$ and for every non-uniform PPT extractor $\mathcal{E}$ it holds that*

$$\Pr\left[r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; (C_0, C_1) := \mathsf{Samp}(1^\lambda, r); x \leftarrow \mathcal{E}(1^\lambda, r) :\ C_0(x) \neq C_1(x)\right] = \mathsf{negl}(\lambda) \ . \quad (2)$$

Definition 7 (Public-Coin diO [IPS15])**.** *A uniform PPT algorithm* diO *is a public-coin differing-input obfuscator for a family of polynomial-size circuits* $\mathcal{C}_\lambda$ *if the following hold:*

- *For all* $\lambda \in \mathbb{N}$, $C \in \mathcal{C}_\lambda$ *and* x: $\Pr\left[\widetilde{C} \leftarrow \mathsf{diO}(1^\lambda, C) : \ C(x) = \widetilde{C}(x)\right] = 1$.
- *For every public-coin differing-input sampler* Samp *for a family of polynomial-size circuits* $\mathcal{C}_\lambda$*, every non-uniform PPT distinguisher* $\mathcal{D}$ *and every* $\lambda \in \mathbb{N}$*:*

$$\begin{aligned}\big|\Pr\left[r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; (C_0, C_1) := \mathsf{Samp}(1^\lambda, r); \tilde{C} \leftarrow \mathsf{diO}(1^\lambda, C_0) : \ 1 \leftarrow \mathcal{D}(r, \tilde{C})\right] - \\ \Pr\left[r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; (C_0, C_1) := \mathsf{Samp}(1^\lambda, r); \tilde{C} \leftarrow \mathsf{diO}(1^\lambda, C_1) : \ 1 \leftarrow \mathcal{D}(r, \tilde{C})\right]\big| = \mathsf{negl}(\lambda) \ .\end{aligned} \tag{3}$$

A candidate $i\mathcal{O}$ was constructed based on a simplified variant of multilinear maps and proven secure in an idealized model [GGH+13b]. The candidate was conjectured to be a $di\mathcal{O}$ for NC^1 [BCP14]. Unfortunately, Garg et al. [GGHW14] present an implausibility result for $di\mathcal{O}$ for arbitrary distributions. However, Ishai et al. [IPS15] argue that current candidate constructions for $i\mathcal{O}$ satisfy their notion of public-coin $di\mathcal{O}$.

2.4 Succinct Non-interactive Arguments of Knowledge

A succinct non-interactive argument of knowledge (SNARK) is a computationally sound non-interactive proof-of-knowledge system for which proofs are universally succinct. A proof π of knowledge of a witness w to a statement η is succinct if the proof length and its verification time are bounded by an a priori fixed polynomial in the statement length $|\eta|$.

Definition 8 (The Universal Relation $\mathcal{R}_\mathcal{U}$ [BG08])**.** *The universal relation* $\mathcal{R}_\mathcal{U}$ *is the set of instance/witness pairs of the form* $((M, m, t), w)$ *where* M *is a TM accepting an input/witness pair* (m, w) *within* t *steps. In particular* $|w| \leq t$.

We define SNARK systems in the common-random-string model following Bitansky et al. [BCCT13, BCC+14, IPS15] as follows.

Definition 9 (SNARK)**.** *A pair of PPT algorithms* $(\mathsf{Prove}, \mathsf{Verify})$ *is a succinct non-interactive argument of knowledge (SNARK) in the common-random-string model for a language* $\mathcal{L}$ *with witness relation* $\mathcal{R} \subseteq \mathcal{R}_\mathcal{U}$ *if there exist polynomials* p, ℓ, q independent *of* $\mathcal{R}$ *such that the following hold:*

1. *Completeness: For every* $(\eta = (M, m, t), w) \in \mathcal{R}$ *the following holds:*

$$\Pr\left[crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; \ \pi \leftarrow \mathsf{Prove}(crs, \eta, w) : \ \mathsf{Verify}(crs, \eta, \pi) = 1\right] = 1 \ .$$

   Moreover, Prove *runs in time* $q(\lambda, |\eta|, t)$.

2. *(Adaptive) Soundness: For every PPT adversary* $\mathcal{A}$*:*

$$\Pr\left[crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; (\eta, \pi) \leftarrow \mathcal{A}(crs) : \ \mathsf{Verify}(crs, \eta, \pi) = 1 \ \wedge \ \eta \notin \mathcal{L}\right] = \mathsf{negl}(\lambda) \ .$$

3. *(Adaptive) Argument of knowledge: For every PPT adversary* $\mathcal{A}$ *there exists a PPT extractor* $\mathcal{E}_\mathcal{A}$ *such that*

$$\Pr\left[\begin{array}{l} crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)} \\ (\eta, \pi) := \mathcal{A}(crs; r); w \leftarrow \mathcal{E}_\mathcal{A}(1^\lambda, crs, r) \end{array} : \begin{array}{r} \mathsf{Verify}(crs, \eta, \pi) = 1 \\ \wedge \ (\eta, w) \notin \mathcal{R} \end{array}\right] = \mathsf{negl}(\lambda) \ .$$

4. *Succinctness: For all* $(crs, \eta, w) \in \{0,1\}^{\mathsf{poly}(\lambda)} \times \mathcal{R}$*, the length of a proof* $\pi \leftarrow \mathsf{Prove}(crs, \eta, w)$ *is bounded by* $\ell(\lambda, \log t)$ *and the running time of* $\mathsf{Verify}(crs, \eta, \pi)$ *is bounded by* $p(\lambda + |\eta|) = p(\lambda + |M| + |m| + \log t)$.

Bitansky et al. [BCC+14] construct SNARKs for $\mathcal{R}_c \subset \mathcal{R}_\mathcal{U}$ where $t = |m|^c$ and c is a constant, based on knowledge-of-exponent assumptions [BCCT13] and extractable collision-resistant hash functions (ECRHF) [BCC+14]. These are both non-falsifiable assumptions, but Gentry and Wichs [GW11] prove that SNARKs cannot be built from falsifiable assumptions via black-box reductions. Relying on exponentially hard one-way functions and ECRHF, [BCC+14] construct SNARKs for $\mathcal{R}_\mathcal{U}$.

3 Constrained PRFs for Unbounded Inputs

As a warm-up we first construct a constrained PRF for sets decided by polynomial-size circuits and show how to extend it to Turing machines in Sect. 3.2.

3.1 A Circuit-Constrained PRF

Our circuit-constrained PRF F uses a puncturable PRF PF with input space $\mathcal{X} = \{0,1\}^n$. The output of $\mathsf{F}(k,x)$ is simply $\mathsf{PF}(k,x)$. To constrain F w.r.t. a circuit C, we construct a circuit $P_{k,C}$, which on input x runs C on x and outputs $\mathsf{PF}(k,x)$ if $C(x) = 1$, and $\perp$ otherwise. A constrained key k_C for C is then an indistinguishability obfuscation of $P_{k,C}$, i.e., $k_C \leftarrow \mathsf{iO}(1^\lambda, P_{k,C})$.

Construction 1 (Circuit-Constrained PRF)**.** *Let $\mathcal{C}_\lambda = \{C\colon \{0,1\}^n \to \{0,1\}\}$ be a family of polynomial-size circuits, $\mathcal{PF}_\lambda = \{\mathsf{PF}\colon \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}\}$ a family of selectively secure puncturable PRFs, and $i\mathcal{O}$ an indistinguishability obfuscator for a family of poly-size circuits $\mathcal{P}_\lambda$ that contains all circuits defined in (4) for all $C \in \mathcal{C}_\lambda$. We construct a family of PRFs $\mathcal{F}_\lambda = \{\mathsf{F}\colon \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}\}$ constrained w.r.t. $\mathcal{C}_\lambda$ with a constrained-key space $\mathcal{K}_\mathcal{C}$ such that $\mathcal{K}_\mathcal{C} \cap \mathcal{K} = \emptyset$.*[2]

$k \leftarrow \mathsf{F.Smp}(1^\lambda)$: Given security parameter λ, output $k \in \mathcal{K}$ as $k \leftarrow \mathsf{PF.Smp}(1^\lambda)$.

$k_C \leftarrow \mathsf{F.Constr}(k, C)$: On input a secret key $k \in \mathcal{K}$ and a description of a circuit $C \in \mathcal{C}_\lambda$, output $k_C \in \mathcal{K}_\mathcal{C}$ as $k_C \leftarrow \mathsf{iO}(1^\lambda, P_{k,C})$, with $P_{k,C} \in \mathcal{P}_\lambda$ defined as:

$$P_{k,C}(x) := \begin{cases} \mathsf{PF}(k,x) & \text{if } |x| = n \wedge C(x) = 1 \\ \perp & \text{otherwise} \end{cases} . \tag{4}$$

$y := \mathsf{F.Eval}(\kappa, x)$: On input $\kappa \in \mathcal{K} \cup \mathcal{K}_\mathcal{C}$ and $x \in \{0,1\}^n$, do the following:

- *If $\kappa \in \mathcal{K}$, output $\mathsf{PF.Eval}(\kappa, x)$.*
- *If $\kappa \in \mathcal{K}_\mathcal{C}$, interpret κ as a circuit and output $\kappa(x)$.*

The proof of selective security of $\mathcal{F}$, as just constructed, is relatively straightforward. Recall that in the selective-security game the adversary $\mathcal{A}$ outputs x^*, then the challenger chooses $k \leftarrow \mathsf{F.Smp}$ and gives $\mathcal{A}$ access to a constrained-key oracle Constr, which can be queried on any C with $C(x^*) = 0$. $\mathcal{A}$ must then distinguish $\mathsf{F}(k, x^*)$ from random. We modify this game by deriving from k a key k_{x^*} which is punctured at x^* and computing constrained keys as obfuscations of $P_{k_{x^*},C}$ (defined like $P_{k,C}$ but using k_{x^*} instead of k). Since $\mathsf{PF}(k,x) = \mathsf{PF}(k_{x^*}, x)$ for all $x \neq x^*$, and since for any circuit C that the adversary can query we have $P_{k,C}(x^*) = P_{k_{x^*},C}(x^*) = \perp$, the circuits $P_{k_{x^*},C}$ and $P_{k,C}$ are functionally equivalent, and thus by Def. 5 the two games are indistinguishable. Note that we also need to ensure that these circuits are of the same size, which can be achieved by appropriate padding.

An adversary $\mathcal{A}$ winning the modified game can be translated into an adversary $\mathcal{B}$ against $\mathcal{PF}$. In the security game for $\mathcal{PF}$ (Fig. 2), $\mathcal{B}$ runs $(x^*, st) \leftarrow \mathcal{A}$ and outputs $(x^*, \{x^*\}, st)$. Given k_{x^*} and y, $\mathcal{B}$ can simulate the modified game and output whatever $\mathcal{A}$ outputs. $\mathcal{B}$'s probability of breaking the security of $\mathcal{PF}$ is the same as that of $\mathcal{A}$ winning the modified game.

[2] W.l.o.g. we assume from now on that $\mathcal{K} \cap \mathcal{K}_\mathcal{C} = \emptyset$, as this can always be achieved by simply prepending a '0' to elements from $\mathcal{K}$ and a '1' to elements from $\mathcal{K}_\mathcal{C}$.

3.2 A TM-Constrained PRF

In this section we construct a family of constrained PRFs for unbounded inputs whose keys can be constrained to sets decided by Turing machines (TM). As a first attempt, in Construction 1 we could replace C in $P_{k,C}$ with a TM M, yielding a TM $P_{k,M}$. We would thus have to use obfuscation for Turing machines rather than just circuits. However, the problem with this construction is that in the proof we would have to replace the underlying PRF key k with a punctured key k_{x^*} for some x^* whose length is not a priori bounded. It is thus not clear how to pad the original key, which could be done in our previous construction.

To overcome this problem we compress the unbounded input to a fixed length by applying a collision-resistant hash function H to it, that is, we evaluate the PRF on hashed inputs. Moreover, we outsource the check of input legitimacy outside the program $P_{k,M}$ by using a SNARK. In particular, when evaluating the PRF, the user computes a SNARK proving that a given hash is the hash value of a legitimate input. The program $P_{k,M}$ is then only given the *hash* of the input to the PRF and a SNARK proof confirming the legitimacy of a preimage, and evaluates the PRF on the hash if the proof verifies.

Note that $P_{k,M}$ can now be implemented by a circuit, which means that we can avoid obfuscation of Turing machines altogether. In our construction a constrained key k_M for a TM M is a public-coin $di\mathcal{O}$ obfuscation of a circuit $P_{k,M}$ which is given (h, π) and checks whether π proves knowledge of an x such that $H(x) = h$ and $M(x) = 1$, and if so, evaluates PF on h.

Let us justify the use of (public-coin) $di\mathcal{O}$ and SNARKs. As for our circuit-constrained PRF, we want to reduce the selective security of the TM-constrained PRF F to the selective security of the underlying puncturable PRF PF. In a first game hop we replace $P_{k,M}$ with $P_{k_{h^*},M}$, which is identical to $P_{k,M}$ except that the key k is replaced with a key k_{h^*} that punctures out $h^* := H(x^*)$. Unfortunately, the two circuits $P_{k,M}$ and $P_{k_{h^*},M}$ are not equivalent: there exists $x \neq x^*$ such that $H(x) = H(x^*)$, and on input $H(x)$, $P_{k,M}$ outputs $\mathsf{PF}(k, H(x)) = \mathsf{PF}(k, h^*)$ and $P_{k_{h^*},M}$ outputs $\perp$. We thus cannot use $i\mathcal{O}$ and hence we use $di\mathcal{O}$ instead. This requires that it be hard to find an input (h, π) on which the two circuits differ, which means that either π proves a wrong statement or it proves knowledge of some x with $H(x) = H(x^*)$. That is, finding a differing input amounts to either breaking soundness of the SNARK or breaking collision-resistance of H. Since both are hard even for adversaries that know the coins used to sample the hash function or the common random string for the SNARK, it suffices to use public-coin $di\mathcal{O}$.

Finally, hash-function collisions are also the reason we need to use SNARKs rather than SNARGs: if an adversary can distinguish obfuscations of $P_{k,M}$ and $P_{k_{h^*},M}$ by finding a collision for H then we need to extract this collision in the security proof, which SNARKs (arguments of *knowledge*) allow us to do.

Definition 10 (R_{legit})**.** *We define the relation $R_{legit} \subset \mathcal{R}_{\mathcal{U}}$, with $\mathcal{R}_{\mathcal{U}}$ defined in Def. 8, to be the set of instance/witness pairs $(((H, M), h, t), x)$ such that $M(x) = 1$ and $H(x) = h$ within t steps, and M is a TM and H is a hash function. We let L_{legit} be the language corresponding to R_{legit}. For notational convenience, we abuse the notation and write $((H, M, h), x) \in R_{legit}$ to mean $(((H, M), h, t), x) \in R_{legit}$ while implicitly setting $t = 2^\lambda$.*

Remark 1. *Let $t = 2^\lambda$ in the definition of R_{legit}; then by succinctness of SNARKs (Def. 9), the length of a SNARK proof is bounded by $\ell(\lambda)$ and its verification time is bounded by $p(\lambda + |M| + |H| + |h|)$, where p, ℓ are a priori fixed polynomials that do not depend on R_{legit}.*

Construction 2 (TM-Constrained PRF)**.** *Let $\mathcal{PF}_\lambda = \{\mathsf{PF} \colon \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}\}$ be a selectively secure puncturable PRF, $\mathcal{H}_\lambda = \{H \colon \{0,1\}^* \to \{0,1\}^n\}_\lambda$ a family of public-coin CR hash functions, diO a public-coin $di\mathcal{O}$ obfuscator for a family of polynomial-size circuits $\mathcal{P}_\lambda$, and SNARK a SNARK system for R_{legit} (cf. Def. 10). We construct a family of PRFs $\mathcal{F}_\lambda = \{\mathsf{F} \colon \mathcal{K} \times \{0,1\}^* \to \mathcal{Y}\}$ constrained w.r.t. to any polynomial-size family of TMs $\mathcal{M}_\lambda$ as follows:*

$\underline{\mathbf{Exp}^{(\emptyset,\{\textsc{Constr},\textsc{Eval}\}),b}_{\mathcal{F},\mathcal{A}}(\lambda)}$

$(x^*, st) \leftarrow \mathcal{A}_1(1^\lambda)$
$K \leftarrow \mathsf{F.Smp}(1^\lambda)$
If $b = 1$
  $y^* := \mathsf{F.Eval}(K, x^*)$
Else
  $y^* \leftarrow \mathcal{Y}$
$b' \leftarrow \mathcal{A}_2^{\textsc{Constr}(\cdot),\textsc{Eval}(\cdot)}(st, y^*)$
Return b'

Oracle $\textsc{Constr}(M)$

If $M \notin \mathcal{M}_\lambda \vee M(x^*) = 1$
  Return $\perp$
$k_M \leftarrow \mathsf{F.Constr}(K, M)$
Return k_M

Oracle $\textsc{Eval}(x)$

If $x = x^*$
  Return $\perp$
$y = \mathsf{F.Eval}(K, x)$
Return y

$\underline{\mathbf{Exp}^{b,(c)}_{\mathcal{F},\mathcal{A}}(\lambda)}$ // $c \in \{0, 1, 2\}$

$(x^*, st) \leftarrow \mathcal{A}_1(1^\lambda)$
$H \leftarrow \mathsf{H.Smp}(1^\lambda)$
$crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$
$k \leftarrow \mathsf{PF.Smp}(1^\lambda)$
$k_{h^*} \leftarrow \mathsf{PF.Constr}(k, \{0,1\}^n \setminus \{H(x^*)\})$
$pp := (H, crs)$
If $b = 1$, $y^* := \mathsf{PF.Eval}(k, H(x^*))$, else $y^* \leftarrow \mathcal{Y}$
$b' \leftarrow \mathcal{A}_2^{\textsc{Constr}(\cdot),\textsc{Eval}(\cdot)}(st, y^*)$
Return b'

Oracle $\textsc{Constr}(M)$

If $M \notin \mathcal{M}_\lambda \vee M(x^*) = 1$
  Return $\perp$
If $c = 0$
  $P := P_{M,H,crs,k}$
    (as defined in (5))
Else
  $P := P_{M,H,crs,k_{h^*}}$
$\widetilde{P} \leftarrow \mathsf{diO}(1^\lambda, P)$
Return $k_M := (M, \widetilde{P}, pp)$

Oracle $\textsc{Eval}(x)$

If $x = x^*$
  Return $\perp$
If $c \leq 1$
  $y := \mathsf{PF.Eval}(k, H(x))$
Else
  If $H(x) = H(x^*)$, abort
  Else $y := \mathsf{PF.Eval}(k_{h^*}, H(x))$
Return y

Figure 3: The original security game and hybrids used in the proof of Theorem 1.

$\underline{K \leftarrow \mathsf{F.Smp}(1^\lambda)}$: *On input a security parameter* λ, *sample* $H \leftarrow \mathsf{H.Smp}(1^\lambda)$, $crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$, $k \leftarrow \mathsf{PF.Smp}(1^\lambda)$, *set* $pp := (H, crs)$) *and return* $K := (k, pp)$.

$\underline{k_M \leftarrow \mathsf{F.Constr}(K, M)}$: *On input* $K = (k, pp = (H, crs))$ *and* $M \in \mathcal{M}_\lambda$, *set*

$$P_{M,H,crs,k}(h, \pi) := \begin{cases} \mathsf{PF.Eval}(k, h) & \text{if } \mathsf{SNARK.Verify}(crs, (H, M, h), \pi) = 1 \\ \perp & \text{otherwise} \end{cases} \tag{5}$$

and compute $\widetilde{P} \leftarrow \mathsf{diO}(1^\lambda, P_{M,H,crs,k})$. *Return* $k_M := (M, \widetilde{P}, (H, crs))$.

$\underline{y := \mathsf{F.Eval}(\kappa, x)}$: *On input* $\kappa \in \mathcal{K} \cup \mathcal{K}_\mathcal{M}$ *and* $x \in \{0,1\}^*$, *do the following:*

- *If* $\kappa \in \mathcal{K}$, $\kappa = (k, (H, crs))$: *return* $\mathsf{PF.Eval}(k, H(x))$.
- *If* $\kappa \in \mathcal{K}_\mathcal{M}$, $\kappa = (M, \widetilde{P}, (H, crs))$: *if* $M(x) = 1$, *let* $h := H(x)$ *(thus* $((H, M, h), x) \in R_{legit}$*), compute* $\pi \leftarrow \mathsf{SNARK.Prove}(crs, (H, M, h), x)$, *interpret* $\widetilde{P}$ *as a circuit and return* $\widetilde{P}(h, \pi)$.

Remark 2. *Note that* $\mathcal{P}_\lambda$ *is in fact a family of circuits with an input length* $n + |\pi|$ *where* $|\pi|$ *is upper bounded by* $\ell(\lambda)$ *even for an exponentially long* x *(cf. Remark 1).*

Theorem 1. $\mathcal{F}_\lambda$ *of Construction 2 is a selectively secure family of constrained PRFs with input space* $\{0,1\}^*$ *for which constrained keys can be derived for any set that can be decided by a polynomial-size Turing machine.*

Proof. Let $\mathcal{A}$ be an arbitrary PPT adversary for game $\mathbf{Exp}^{(\emptyset,\{\textsc{Constr},\textsc{Eval}\}),b}_{\mathcal{F},\mathcal{A}}(\lambda)$, as defined in Fig. 3, which we abbreviate as $\mathbf{Exp}^b$ for simplicity. We need to show that $\mathbf{Exp}^0$ and $\mathbf{Exp}^1$ are indistinguishable. Our proof will be by game hopping and we define a series of hybrid games

$\mathbf{Exp}^{b,(0)} := \mathbf{Exp}^{b}$, $\mathbf{Exp}^{b,(1)}$ and $\mathbf{Exp}^{b,(2)}$, which are all defined in Fig. 3. We show that for $b = 0, 1$ and $c = 0, 1$ the games $\mathbf{Exp}^{b,(c)}$ and $\mathbf{Exp}^{b,(c+1)}$ are indistinguishable and that $\mathbf{Exp}^{0,(2)}$ and $\mathbf{Exp}^{1,(2)}$ are also indistinguishable, which concludes the proof.

$\mathbf{Exp}^{b,(0)}$ is the original game $\mathbf{Exp}^{b,(\emptyset,\{\textsc{Constr},\textsc{Eval}\})}_{\mathcal{F},\mathcal{A}}(\lambda)$ for Construction 2.

$\mathbf{Exp}^{b,(1)}$ differs from $\mathbf{Exp}^{b,(0)}$ by replacing the full key of the puncturable PRF PF, with one that is punctured at $H(x^*)$ in the definition of P.

$\mathbf{Exp}^{b,(2)}$ differs from $\mathbf{Exp}^{b,(1)}$ by answering $\textsc{Eval}$ queries using the punctured key k_{h^*} and aborting whenever the query is a collision with x^* for H.

The only difference between $\mathbf{Exp}^{b,(0)}$ and $\mathbf{Exp}^{b,(1)}$ is the definition of the circuits P that are obfuscated when the $\textsc{Constr}$ oracle is queried. In $\mathbf{Exp}^{b,(0)}$ the circuit P is defined as in (5), with $k \leftarrow \mathsf{PF.Smp}(1^\lambda)$. In $\mathbf{Exp}^{b,(1)}$, the key k is replaced by $k_{h^*} \leftarrow \mathsf{PF.Constr}(k, \{0,1\}^n \setminus \{H(x^*)\})$, a key that punctures out $H(x^*)$. By a hybrid argument there must exist some query (say the ith for M_i) where the adversary distinguishes a $di\mathcal{O}$ obfuscation of $P_{M_i,H,crs,k}$ from one of $P_{M_i,H,crs,k_{h^*}}$. Thus, there exists a $di\mathcal{O}$ extractor that extracts an input $(\hat{h}, \hat{\pi})$ on which $P_{M_i,H,crs,k}$ and $P_{M_i,H,crs,k_{h^*}}$ differ.

By correctness of PF, the circuits only differ on inputs $(\hat{h}, \hat{\pi})$, where

$$\hat{h} = H(x^*) \ , \tag{6}$$

as that is where the punctured key behaves differently. Moreover, the extracted proof $\hat{\pi}$ must be valid for $(H, M_i, \hat{h})$, as otherwise both circuits output $\perp$. By SNARK extractability, we can extract a witness $\hat{x}$ for $(H, M_i, \hat{h}) \in L_{legit}$, that is, (i) $M_i(\hat{x}) = 1$ and (ii) $H(\hat{x}) = \hat{h}$. Since M_i is a legitimate query, we have $M_i(x^*) = 0$, which together with (i) implies $\hat{x} \neq x^*$. On the other hand, (ii) and (6) imply $H(\hat{x}) = H(x^*)$. Together, this means $(\hat{x}, x^*)$ is a collision for H.

Proposition 1. *For $b = 0, 1$, $\mathbf{Exp}^{b,(0)}$ and $\mathbf{Exp}^{b,(1)}$ are computationally indistinguishable if diO is a public-coin differing-input obfuscator and $\mathcal{H}$ is public-coin collision-resistant.*

For the game hop from games $\mathbf{Exp}^{b,(1)}$ to $\mathbf{Exp}^{b,(2)}$, indistinguishability follows directly from collision resistance of $\mathcal{H}$, as the only difference is that $\mathbf{Exp}^{b,(2)}$ aborts when $\mathcal{A}$ finds a collision.

Proposition 2. *For $b = 0, 1$, $\mathbf{Exp}^{b,(1)}$ and $\mathbf{Exp}^{b,(2)}$ are computationally indistinguishable for if $\mathcal{H}$ is collision-resistant.*

We have now reached a game, $\mathbf{Exp}^{b,(2)}$, in which the key k is only used to create a punctured key k_{h^*}. The experiment can thus be simulated by an adversary $\mathcal{B}$ against selective security of $\mathcal{PF}$, which first asks for a key for the set $\{0,1\}^n \setminus \{H(x^*)\}$ and then uses $\mathcal{A}$ to distinguish $y^* = \mathsf{PF.Eval}(k, H(x^*))$ from random.

Proposition 3. $\mathbf{Exp}^{0,(2)}$ *and* $\mathbf{Exp}^{1,(2)}$ *are indistinguishable if $\mathcal{PF}$ is a selectively secure family of puncturable PRFs.*

Theorem 1 now follows from Propositions 1, 2 and 3, whose proofs can be found in Appendices A.1, A.2 and A.3. □

4 Applications

As a first application of TM-constrained PRFs we construct the first identity-based non-interactive key-agreement protocol with no a priori bound on the number of parties agreeing on a shared key. Our second application is a broadcast encryption scheme where during setup the number of potential receivers need not be known.

4.1 ID-Based Non-interactive Key Exchange for Unbounded Groups

Identity-based non-interactive key exchange (ID-NIKE) [SOK00] allows users to compute shared keys without any interaction—it suffices to know the identity of the users one wants to share a key with. In our construction a user can compute a shared key for any group of users and there is no a priori bound on the size of these groups. We generalize the construction of [BW13, HKKW14], in which identities are elements from $\{0,1\}^\ell$ and the system is set up by creating a secret key msk for a constrained PRF. A key for a group of users $\{id_1, \ldots, id_n\}$ is defined as $\mathsf{F.Eval}(msk, x)$, where $x = id_1 \| \ldots \| id_n$ and we assume identities are always ordered lexicographically. Boneh and Zhandry [BZ14] suggest a similar scheme (assuming indistinguishability obfuscation) which does not require any setup, but where each party publishes a value, and a key for a set of parties can be computed when knowing this values for each party in the set.

Since in the previous constructions the CPRF is set up for a fixed input length m there is an a-prior-fixed maximum number of users that can share a key, namely m/ℓ. As a user's id could appear in any position of the string x, the owner of id is given constrained keys for the sets $id\|?^{(n-1)\ell} := \{id\|z \,|\, z \in \{0,1\}^{(n-1)\cdot\ell}\}$, $?^\ell\|id\|?^{(n-2)\ell}$, $\ldots$, $?^{(n-1)\ell}\|id$. These keys then allow the user to compute the key for any set which she is part of.

We generalize this to sets of users of unbounded size. Again, a key for a (lexicographically ordered) set $\{id_1, \ldots, id_n\}$ is defined as $\mathsf{F.Eval}(msk, id_1\| \ldots \| id_n)$, but now n can be arbitrary and is not fixed in advance. In order to let a user with identity id compute the keys of the sets which she is part of—but not anything else—, she is given a constrained key for the following Turing machine M_{id}: on input $x \in \{0,1\}^*$, machine M_{id} outputs 1 if and only if id is a substring of x, which starts at position $i \cdot \ell + 1$, for some $i \geq 0$, that is, at position 1 or $\ell + 1$ or $2\ell + 1$, etc.

ID-NIKE. An (unbounded) ID-NIKE scheme consists of three algorithms:

- $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda)$: On input λ, output public parameters pp and a master secret key msk.
- $sk_{id} \leftarrow \mathsf{Extract}(msk, id)$: On input msk and $id \in \{0,1\}^\ell$, output a secret key sk_{id}.
- $k_{\mathcal{I}} \leftarrow \mathsf{KeyGen}(pp, sk_{id}, \mathcal{I})$: On input pp, a key sk_{id} for id and a list $\mathcal{I} \subseteq \{0,1\}^\ell$ of n (for arbitrary n) users with $id \in \mathcal{I}$, output a shared key $k_{\mathcal{I}}$.

Correctness is defined as follows: for all $id, id' \in \{0,1\}^\ell$, all $\mathcal{I} \subseteq \{0,1\}^\ell$ with $id, id' \in \mathcal{I}$, all $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda)$, $sk_{id} \leftarrow \mathsf{Extract}(msk, id)$ and $sk_{id'} \leftarrow \mathsf{Extract}(msk, id')$, it holds that $\mathsf{KeyGen}(pp, sk_{id}, \mathcal{I}) = \mathsf{KeyGen}(pp, sk_{id'}, \mathcal{I})$.

Following [PS09], we define security via a game where an adversary can obtain secret keys sk_{id} for identities of his choice and can query secret keys $k_{\mathcal{I}}$ for sets $\mathcal{I}$ of his choice. The scheme is secure if the adversary cannot distinguish a key $k_{\mathcal{I}^*}$ for a set $\mathcal{I}^*$ of his choice from random, where we must have $id \notin \mathcal{I}^*$ for all id for which the adversary queried key extraction, and $\mathcal{I}^* \neq \mathcal{I}$ for all $\mathcal{I}$ for which the adversary queried a shared key. We prove that our scheme satisfies the selective variant of this definition, where the adversary must output $\mathcal{I}^*$ before getting access to its oracles.

ID-NIKE from constrained PRFs for unbounded inputs. Our unbounded ID-NIKE is obtained from a TM-constrained PRF $(\mathsf{F.Smp}, \mathsf{F.Constr}, \mathsf{F.Eval})$ as follows.

- $\mathsf{Setup}(1^\lambda)$: Return $msk \leftarrow \mathsf{F.Smp}(1^\lambda)$. (There are no parameters pp.)
- $\mathsf{Extract}(msk, id)$: On input $id \in \{0,1\}^\ell$ do the following: define a Turing machine M_{id} that on input a string $x \in \{0,1\}^*$ outputs 1 iff x is of the form $x'\|id\|x''$ with $x' \in \{0,1\}^{n'\cdot\ell}$ and $x'' \in \{0,1\}^{n''\cdot\ell}$ for some $n', n'' \in \mathbb{N}$; return $sk_{id} \leftarrow \mathsf{F.Constr}(msk, M_{id})$.
- $\mathsf{KeyGen}(sk_{id}, \mathcal{I})$: If $\mathcal{I} = \{id_1, \ldots, id_n\} \subseteq \{0,1\}^\ell$ for some n and $id \in \mathcal{I}$ then define $x := id_{i_1}\| \ldots \| id_{i_n}$, with $id_{i_j} < id_{i_j+1}$ for all j, and output $k_{\mathcal{I}} := \mathsf{F.Eval}(sk_{id}, x)$; else output $\perp$.

Correctness of our scheme follows from correctness of the underlying constrained PRF. Selective security of the ID-NIKE follows from selective security of the CPRF (Def. 2). Given an adversary $\mathcal{A}$ against the ID-NIKE, we construct an adversary $\mathcal{B}$ against the CPRF. First $\mathcal{B}$ runs $\mathcal{A}$ to obtain $\mathcal{I}^*$ and sends x^*, the concatenation of the lexicographically ordered elements of $\mathcal{I}^*$, to its challenger.

$\mathcal{B}$ answers $\mathcal{A}$'s queries as follows: When $\mathcal{A}$ queries a secret key for $id \in \mathcal{I}^*$ or the shared key for $\mathcal{I}^*$ then reply with $\perp$. On a legal secret-key query for id, construct a Turing machine M_{id} as in the definition of $\mathsf{Extract}$, query the $\textsc{Constr}$ oracle on M_{id} and forward the reply to $\mathcal{A}$. When $\mathcal{A}$ queries a shared key for a set $\mathcal{I} \neq \mathcal{I}^*$, construct x as in KeyGen, query $\textsc{Eval}$ on x and forward the reply.

Note that $\mathcal{B}$ makes no illegal queries (any queried M evaluates x^* to 0 and x^* is never queried to $\textsc{Eval}$) and perfectly simulates the game for $\mathcal{A}$. When $\mathcal{B}$ receives a value y which is either $\mathsf{F.Eval}(msk, x^*)$ or random, it forwards y as the challenge key $k_{\mathcal{I}^*}$ to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ returns. $\mathcal{B}$ thus breaks the CPRF with the same probability as $\mathcal{A}$ breaking the ID-NIKE, which concludes the proof.

4.2 Broadcast Encryption to Unbounded Number of Users

As a second application we use a constrained PRF for unbounded inputs to construct a broadcast encryption (BE) [FN94] where there is no limit on the number of receivers. We start with defining *dynamic* BE, which allows users to join the system after it is set up. Each user is identified by a consecutive number i.

A broadcast encryption scheme $\mathcal{BE}$ for a symmetric encryption scheme $(\mathsf{enc}, \mathsf{dec})$ with key space $\mathcal{K}_{\text{sym}}$ consists of the following four PPT algorithms:

- $(bk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$: On input a security parameter λ, output a broadcast key bk and a master secret key msk, used to enroll new members in the system.
- $sk_i \leftarrow \mathsf{KeyGen}(msk, i)$: On input a master key msk and a member identity i, output sk_i, a secret key for member i.
- $(hdr, K) \leftarrow \mathsf{Encrypt}(bk, S)$: On input a set $S \subseteq \mathbb{N}$ and a broadcast key bk, output a header hdr and a key $K \in \mathcal{K}_{\text{sym}}$. (A message m is then broadcast as $(S, hdr, \mathsf{enc}(K, m))$.)
- $K \leftarrow \mathsf{Decrypt}(i, sk_i, S, hdr)$: On input a member identity i, an associated secret key sk_i, a set $S \subseteq \mathbb{N}$ and a header hdr, if $i \in S$ then output a symmetric key $K \in \mathcal{K}_{\text{sym}}$. (A broadcast (S, hdr, C) can then be decrypted via $m \leftarrow \mathsf{dec}(K, C)$.)

Like Boneh and Waters [BW13], whose construction we build on, we will construct a *secret-key* BE scheme, where bk must only be known to the broadcaster. Correctness of a BE scheme is defined as follows: for all $S \subseteq \mathbb{N}$, $i \in S$, all $(bk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$, $sk_i \leftarrow \mathsf{KeyGen}(msk, i)$ and $(hdr, K) \leftarrow \mathsf{Encrypt}(bk, S)$, we have $K \leftarrow \mathsf{Decrypt}(i, sk_i, S, hdr)$.

Selective security is defined via the following game $\mathbf{Exp}^{\text{BE-}b}$ for an adversary $\mathcal{A}$:

$\underline{\mathbf{Exp}^{\text{BE-}b}_{\mathcal{BE},\mathcal{A}}(\lambda)}$
$(bk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$
$(S^*, st) \leftarrow \mathcal{A}_1(1^\lambda)$
$(hdr^*, K^*) \leftarrow \mathsf{Encrypt}(bk, S^*)$
If $b = 0$ then $K^* \leftarrow \mathcal{K}_{\text{sym}}$
$b' \leftarrow \mathcal{A}_2^{\textsc{Key}(\cdot),\textsc{Encrypt}(\cdot)}(st, (hdr^*, K^*))$
Return b'

$\underline{\textbf{Oracle } \textsc{Key}(i)}$
If $i \in S^*$
  Return $\perp$
$sk_i \leftarrow \mathsf{KeyGen}(msk, i)$
Return sk_i

$\underline{\textbf{Oracle } \textsc{Encrypt}(S)}$
If $S = S^*$
  Return $\perp$
$(hdr, K) \leftarrow \mathsf{Encrypt}(bk, S)$
Return (hdr, K)

We say $\mathcal{BE}$ is secure if

$$\mathsf{Adv}^{\text{BE}}_{\mathcal{BE},\mathcal{A}}(\lambda) := \left| \Pr[\mathbf{Exp}^{\text{BE-0}}_{\mathcal{BE},\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\text{BE-1}}_{\mathcal{BE},\mathcal{A}}(\lambda) = 1] \right| = \mathsf{negl}(\lambda).$$

BE from constrained PRFs for unbounded inputs. For a finite set $S \subseteq \mathbb{N}$, we define the characteristic vector χ_S as the binary vector whose length equals the largest element in S and whose entry at position i is 1 iff $i \in S$. Let $(\mathsf{enc}, \mathsf{dec})$ be a symmetric encryption scheme with key space $\mathcal{K}_{\text{sym}}$. Let $\mathcal{F} = \{\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}\}$ be a constrained PRF with input space $\mathcal{X} = \{0,1\}^*$ and range $\mathcal{Y} = \mathcal{K}_{\text{sym}}$ for which constrained keys k_i for the following set can be computed:

$$S_i := \{x \in \{0,1\}^* \mid x_i = 1\} \ . \tag{7}$$

(As S_i can be decided by a polynomial-time Turing machine, our construction from Sect. 3.2 can be used.) We define a broadcast encryption scheme $\mathcal{BE}$ with *optimal ciphertext length* (that is, the header is empty: $hdr = \emptyset$) as follows:

- $\mathsf{Setup}(1^\lambda)$: Generate $k \leftarrow \mathsf{F.Smp}(1^\lambda)$ and return $bk := k$, $msk := k$.
- $\mathsf{KeyGen}(msk, i)$: Return $k_i \leftarrow \mathsf{F.Constr}(msk, S_i)$ with S_i as in (7).
- $\mathsf{Encrypt}(bk, S)$: Let $\chi_S \in \{0,1\}^*$ be the characteristic vector of S; compute $K \leftarrow \mathsf{F.Eval}(bk, \chi_S)$ and output $(\emptyset, K)$.
- $\mathsf{Decrypt}(i, sk_i, S, hdr)$: With χ_S as above, output $K \leftarrow \mathsf{F.Eval}(sk_i, \chi_S)$.

Correctness of $\mathcal{BE}$ follows from correctness of $\mathcal{F}$; security follows by reduction to selective pseudorandomness of $\mathcal{F}$. Let $\mathcal{A}$ be a PPT adversary that breaks security of $\mathcal{BE}$; then we construct a PPT algorithm $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that breaks $\mathcal{F}$ with the same probability:

$\underline{\mathcal{B}_1(1^\lambda)}$

- $(S^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(1^\lambda)$.
- Let x^* be the characteristic string of S^*.
- Return $(x^*, st_{\mathcal{A}})$.

$\underline{\mathcal{B}_2^{\textsc{Constr}(\cdot),\textsc{Eval}(\cdot)}(st, K^*)}$

- $b' \leftarrow \mathcal{A}_2^{\textsc{Key}(\cdot),\textsc{Encrypt}(\cdot)}(st, (\emptyset, K^*))$;
  - simulate $\textsc{Key}(i)$: define S_i as in (7); **query** $k_i \leftarrow \textsc{Constr}(S_i)$; reply k_i;
  - simulate $\textsc{Encrypt}(S)$: define the characteristic vector $\chi_S \in \{0,1\}^*$ of S; **query** $K \leftarrow \textsc{Eval}(\chi_S)$; reply $(\emptyset, K)$.
- Return b'.

By construction, we have $\mathbf{Exp}_{\mathcal{F},\mathcal{B}}^{(\emptyset,\{\textsc{Constr},\textsc{Eval}\}),b} = \mathbf{Exp}_{\mathcal{BE},\mathcal{A}}^{\text{BE-}b}$, which proves the claim.

References

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. http://eprint.iacr.org/2013/689.

[BCC+14] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. *IACR Cryptology ePrint Archive*, 2014:580, 2014.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 111–120. ACM Press, June 2013.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 52–73. Springer, Heidelberg, February 2014.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM J. Comput.*, 38(5):1661–1694, 2008.

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

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275. Springer, Heidelberg, August 2005.

[BH08] Dan Boneh and Michael Hamburg. Generalized identity based and broadcast encryption schemes. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 455–470. Springer, Heidelberg, December 2008.

[BST14] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 102–121. Springer, Heidelberg, December 2014.

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

[BWZ14] Dan Boneh, Brent Waters, and Mark Zhandry. Low overhead broadcast encryption from multilinear maps. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 206–223. Springer, Heidelberg, August 2014.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Heidelberg, August 2014.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 476–493. Springer, Heidelberg, August 2013.

[FHPS13] Eduarda S. V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 513–530. Springer, Heidelberg, August 2013.

[FKPR14] Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained PRFs. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 82–101. Springer, Heidelberg, December 2014.

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 480–491. Springer, Heidelberg, August 1994.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, Heidelberg, May 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 518–535. Springer, Heidelberg, August 2014.

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

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

[HKKW14] Dennis Hofheinz, Akshay Kamath, Venkata Koppula, and Brent Waters. Adaptively secure constrained pseudorandom functions. Cryptology ePrint Archive, Report 2014/720, 2014. http://eprint.iacr.org/.

[HKW14] Susan Hohenberger, Venkata Koppula, and Brent Waters. Adaptively secure puncturable pseudorandom functions in the standard model. Cryptology ePrint Archive, Report 2014/521, 2014. http://eprint.iacr.org/2014/521.

[HR04] Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 92–105. Springer, Heidelberg, August 2004.

[IPS15] Yuval Ishai, Omkant Pandey, and Amit Sahai. Public-coin differing-inputs obfuscation and its applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 668–697. Springer, Heidelberg, March 2015.

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 419–428. ACM Press, June 2015.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 669–684. ACM Press, November 2013.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 239–256. Springer, Heidelberg, May 2014.

[PPS11] Duong Hieu Phan, David Pointcheval, and Mario Strefler. Security notions for broadcast encryption. In Javier Lopez and Gene Tsudik, editors, *ACNS 11*, volume 6715 of *LNCS*, pages 377–394. Springer, Heidelberg, June 2011.

[PS09] Kenneth G. Paterson and Sriramkrishnan Srinivasan. On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. *Des. Codes Cryptography*, 52(2):219–241, 2009.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 500–517. Springer, Heidelberg, August 2014.

[SOK00] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *SCIS 2000*, Okinawa, Japan, January 2000.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.

[YFDL04] Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 04*, pages 354–363. ACM Press, October 2004.

A Proofs

A.1 Proof of Proposition 1

Let $b \in \{0,1\}$ be arbitrarily fixed. Assume towards contradiction that there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that distinguishes $\mathbf{Exp}^{b,(0)}$ and $\mathbf{Exp}^{b,(1)}$ with non-negligible probability, i.e., there exists a polynomial $p(\cdot)$ such that for infinitely many λ

$$\left| \Pr[\mathbf{Exp}^{b,(0)}_{\mathcal{F},\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{b,(1)}_{\mathcal{F},\mathcal{A}}(\lambda) = 1] \right| \geq \tfrac{1}{p(\lambda)} \quad . \tag{8}$$

We use $\mathcal{A}$ to construct a series of PPT adversaries $\mathcal{D}^{(1)}$, $\mathcal{D}^{(2)}$, ..., one of which non-negligibly distinguishes public-coin $di\mathcal{O}$-obfuscations.

Concretely, we construct a series of hybrids between $\mathbf{Exp}^{b,(0)}$ and $\mathbf{Exp}^{b,(1)}$ as follows. Let $q = q(\lambda)$ be a polynomial upper bound on the total number of constraining queries $\mathcal{A}$ makes. Define the i-th hybrid $\mathbf{Exp}^{b,(0,i)}$ like $\mathbf{Exp}^{b,(0)}$, except that the first i constraining queries are answered by using the punctured key k_{h^*} of the puncturable PRF, and all remaining queries are answered by using private key k. By construction, we have $\mathbf{Exp}^{b,(0,0)} = \mathbf{Exp}^{b,(0)}$ and $\mathbf{Exp}^{b,(0,q)} = \mathbf{Exp}^{b,(1)}$.

We use $\mathcal{A}$ to construct a PPT adversary $\mathcal{D}^{(i)}$ which distinguishes public-coin $di\mathcal{O}$-obfuscations. We define the series of public-coin differing-input samplers $\mathsf{Samp}^{(i)}$ as follows:

$\underline{\mathsf{Samp}^{(i)}(1^\lambda, r := r_{\mathcal{A}} \| r_H \| r_s \| r_k \| r_c \| r_y \| r_1 \| \ldots \| r_{i-1})}$ // $r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.

- $(x^*, st_{\mathcal{A}}) := \mathcal{A}_1(1^\lambda; r_{\mathcal{A}})$.
- $H := \mathsf{H.Smp}(1^\lambda; r_H)$ and $h^* := H(x^*)$.
- $crs := r_s$.
- $k := \mathsf{PF.Smp}(1^\lambda; r_k)$.
- $k_{h^*} := \mathsf{PF.Constr}(k, \{0,1\}^n \setminus \{h^*\}; r_c)$.
- If $b = 1$ then $y^* := \mathsf{PF.Eval}(k, H(x^*))$
- Else $y^* \leftarrow \mathcal{Y}$, using randomness r_y.
- $b' \leftarrow \mathcal{A}_2^{\textsc{Constr}(\cdot),\textsc{Eval}(\cdot)}(st_{\mathcal{A}}, y^*; r_{\mathcal{A}})$;
  - simulate the j-th query $\textsc{Constr}(M_j)$:
    - if $M_j \notin \mathcal{M}_\lambda \vee M_j(x^*) = 1$, reply $\perp$;
    - if $j < i$, compute $P = P_{M_j,H,crs,k_{h^*}}$ as defined in (5) and $\widetilde{P} := \mathsf{diO}(1^\lambda, P; r_j)$ and return $k_M := (M_j, \widetilde{P}, (H, crs))$.
    - if $j = i$, stop and output $(P_0 := P_{M_i,H,crs,k}, P_1 := P_{M_i,H,crs,k_{h^*}})$.
  - simulate $\textsc{Eval}(x)$:
    if $x = x^*$, reply $\perp$; else reply $y := \mathsf{PF.Eval}(k, H(x))$.

We now define a public-coin $di\mathcal{O}$ distinguisher $\mathcal{D}^{(i)}$ as follows:

$\underline{\mathcal{D}^{(i)}(r, \widetilde{P})}$ // $\widetilde{P}$ is either $\widetilde{P}_0 \leftarrow \mathsf{diO}(1^\lambda, P_0)$ or $\widetilde{P}_1 \leftarrow \mathsf{diO}(1^\lambda, P_1)$.

- Using randomness r run all steps of $\mathsf{Samp}^{(i)}$.
- Answer $\mathcal{A}$'s i-th $\textsc{Constr}$ query for M_i with $(M_i, \widetilde{P})$.
- Simulate the remaining $\textsc{Constr}(M_j)$ as follows:
  - if $M_j \notin \mathcal{M}_\lambda \vee M_j(x^*) = 1$, reply $\perp$;
  - compute $P_j = P_{M_j,H,crs,k}$ as defined in (5) and $\widetilde{P}_j \leftarrow \mathsf{diO}(1^\lambda, P_j)$;
  - return $k_M := (M_j, \widetilde{P}_j, (H, crs))$.
- When $\mathcal{A}_2$ outputs b', output b'.

If $\widetilde{P}$ is a public-coin $di\mathcal{O}$ obfuscation of P_0 then $\mathcal{D}^{(i)}$ simulates $\mathbf{Exp}^{b,(0,i-1)}$ for $\mathcal{A}$, while if $\widetilde{P}$ is an obfuscation of P_1 then it simulates $\mathbf{Exp}^{b,(0,i)}$ for $\mathcal{A}$. If $\mathcal{A}$ distinguishes $\mathbf{Exp}^{b,(0)} = \mathbf{Exp}^{b,(0,0)}$ from $\mathbf{Exp}^{b,(1)} = \mathbf{Exp}^{b,(0,q)}$ with probability $1/p(\lambda)$, then by a hybrid argument for some i, $\mathcal{A}$

distinguishes $\mathbf{Exp}^{b,(0,i-1)}$ and $\mathbf{Exp}^{b,(0,i)}$ with probability $1/(p(\lambda)q(\lambda))$. Thus infinitely many λ, it holds that

$$\frac{1}{p(\lambda)q(\lambda)} \leq \left|\Pr[\mathbf{Exp}_{\mathcal{A}}^{b,(0,i)}(\lambda)=1] - \Pr[\mathbf{Exp}_{\mathcal{A}}^{b,(0,i-1)}(\lambda)=1]\right| = \\ \Big|\Pr\left[r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; (P_0,P_1) := \mathsf{Samp}(1^\lambda, r); \widetilde{P}_0 \leftarrow \mathsf{diO}(1^\lambda, P_0) : \ 1 \leftarrow \mathcal{D}(r, \widetilde{P}_0)\right] - \\ \Pr\left[r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; (P_0,P_1) := \mathsf{Samp}(1^\lambda, r); \widetilde{P}_1 \leftarrow \mathsf{diO}(1^\lambda, P_1) : \ 1 \leftarrow \mathcal{D}(r, \widetilde{P}_1)\right]\Big| \ .$$

By security of $di\mathcal{O}$ (Def. 7), this means that $\mathsf{Samp}^{(i)}$ cannot satisfy Def. 6. Thus, there exists a non-uniform PPT extractor $\mathcal{E}^{(i)}$ that given r finds an input $\chi = (h, \pi)$ on which P_0 and P_1, output by $\mathsf{Samp}^{(i)}$ on coins r, differ. In particular, for some polynomial $\ell(\cdot)$ and infinitely many λ it holds that

$$\Pr\left[\begin{array}{l} r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; \\ (P_0,P_1) \leftarrow \mathsf{Samp}^{(i)}(1^\lambda, r); \chi \leftarrow \mathcal{E}^{(i)}(r) \end{array} : \ P_0(\chi) \neq P_1(\chi)\right] \geq \frac{1}{\ell(\lambda)} \ . \tag{9}$$

Let $\hat{\chi} = (\hat{h}, \hat{\pi})$ be a differing input output by $\mathcal{E}^{(i)}$. Recall that $\hat{\pi}$ is a short proof of $\hat{\eta} = (H, M_i, \hat{h}) \in L_{legit}$, i.e., a short proof of knowledge of a witness x such that $M_i(x) = 1$ and $H(x) = \hat{h}$. By the definition of $\mathsf{Samp}^{(i)}$ we have $P_0 := P_{M_i,H,crs,k}$ and $P_1 := P_{M_i,H,crs,k_{h^*}}$, as defined in (5). Since $(\hat{h}, \hat{\pi})$ is a differing input, the following conditions must hold.

$\mathsf{condition}(1)$: $\mathsf{SNARK.Verify}(crs, (H, M_i, \hat{h}), \hat{\pi}) = 1$, for otherwise both P_0 and P_1 output $\perp$.

$\mathsf{condition}(2)$: $\hat{h} = h^* = H(x^*)$, for otherwise P_0 outputs $\mathsf{PF.Eval}(k, \hat{h})$ and P_1 outputs $\mathsf{PF.Eval}(k_{h^*}, \hat{h})$, which are equal by the correctness of puncturing.

$\mathsf{condition}(3)$: $M_i(x^*) = 0$; otherwise the Constr query would have returned $\perp$.

Since the SNARK $\hat{\pi}$ extracted by $\mathcal{E}^{(i)}$ is a proof of knowledge, we can extract a witness $\hat{x}$ for it. In order to formally apply item 3. of Def. 9, we first construct a machine $\mathcal{A}_{\mathsf{snrk}}$ that outputs $\hat{\pi}$ together with the statement. $\mathcal{A}_{\mathsf{snrk}}$ simply runs $\mathcal{E}^{(i)}$ except that it embeds the *crs* from its input into the randomness.

$\underline{\mathcal{A}_{\mathsf{snrk}}(crs)}$
- Sample randomness $r_{\mathcal{A}}$, r_H, r_k, r_c, r_y, r_1, $\ldots$, r_{i-1}
- Set $r := r_{\mathcal{A}} \| r_H \| crs \| r_k \| r_c \| r_y \| r_1 \| \ldots \| r_{i-1}$.
- Simulate $\mathsf{Samp}^{(i)}(r)$;
  - let H be the sampled hash function.
  - let M_i be the i-th Constr query.
- $(\hat{h}, \hat{\pi}) \leftarrow \mathcal{E}^{(i)}(crs, r)$.
- Output $(\eta := (H, M_i, \hat{h}), \hat{\pi})$.

By the construction of $\mathcal{A}_{\mathsf{snrk}}$, Eq. (9) and $\mathsf{condition}(1)$ we have that

$$\Pr\left[crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; (\eta, \hat{\pi}) \leftarrow \mathcal{A}_{\mathsf{snrk}}(crs) : \ \mathsf{Verify}(crs, \eta, \hat{\pi}) = 1\right] \geq \frac{1}{\ell(\lambda)} \ . \tag{10}$$

Further, since SNARK is an adaptive argument of knowledge, there exists $\mathcal{E}_{\mathcal{A}_{\mathsf{snrk}}}$ which extracts a witness, that is:

$$\Pr\left[\begin{array}{l} crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)} \\ (\eta, \hat{\pi}) := \mathcal{A}_{\mathsf{snrk}}(crs; r); \hat{x} \leftarrow \mathcal{E}_{\mathcal{A}_{\mathsf{snrk}}}(crs, r) \end{array} : \begin{array}{c} \mathsf{Verify}(crs, \eta, \hat{\pi}) = 1 \\ \wedge \ (\eta, \hat{x}) \notin R_{legit} \end{array}\right] = \mathsf{negl}(\lambda) \ ,$$

which together with (10) yields:

$$\Pr\left[\begin{array}{l} crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)} \\ (\eta, \hat{\pi}) := \mathcal{A}_{\mathsf{snrk}}(crs; r); \hat{x} \leftarrow \mathcal{E}_{\mathcal{A}_{\mathsf{snrk}}}(crs, r) \end{array} : \quad (\eta, \hat{x}) \in R_{legit} \right] \geq \frac{1}{\ell(\lambda)} - \mathsf{negl}(\lambda) \ . \tag{11}$$

We now construct an adversary $\mathcal{A}_{\mathsf{cll\text{-}fnd}}$ against $\mathcal{H}$ that on input λ and coins r_H, used to sample a function H, outputs a collision for H: Given r_H, $\mathcal{A}_{\mathsf{cll\text{-}fnd}}$ generates a CRS for SNARKs, then runs $\mathcal{A}_{\mathsf{snrk}}(crs)$, but using the randomness r_H from its input, and then runs $\mathcal{E}_{\mathcal{A}_{\mathsf{snrk}}}$ to extract a collision:

$\underline{\mathcal{A}_{\mathsf{cll\text{-}fnd}}(1^\lambda, r_H^*)}$

- $crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
- Sample randomness $r_{\mathcal{A}}$, r_H, r_k, r_c, r_y, r_1, $\ldots$, r_{i-1}
- Set $r := r_{\mathcal{A}} \| r_H^* \| crs \| r_k \| r_c \| r_y \| r_1 \| \ldots \| r_{i-1}$.
- $(\hat{h}, \hat{\pi}) \leftarrow \mathcal{E}^{(i)}(r)$.
- Set $r' := r_{\mathcal{A}} \| r_H^* \| r_k \| r_c \| r_y \| r_1 \| \ldots \| r_{i-1}$ // $\mathcal{A}_{\mathsf{snrk}}$'s coins
- $\hat{x} \leftarrow \mathcal{E}_{\mathcal{A}_{\mathsf{snrk}}}(crs, r')$.
- Output $(\hat{x}, x^*)$ as a collision pair for H.

By (11), with non-negligible probability, the values $M_i, \hat{h}, \hat{\pi}$ computed during the execution of $\mathcal{A}_{\mathsf{cll\text{-}fnd}}$ satisfy $((H, M_i, \hat{h}), \hat{x}) \in R_{legit}$, that is, $M_i(\hat{x}) = 1$ and $\hat{h} = H(\hat{x})$. By $\mathsf{condition}(3)$, $M_i(x^*) = 0$, and hence $\hat{x} \neq x^*$. By $\mathsf{condition}(2)$, $\hat{h} = H(x^*)$, and hence (x, x^*) is a collision. In particular, the following is non-negligible:

$$\Pr\left[\begin{array}{l} r_H \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}; H := \mathsf{HSmp}(1^\lambda, r_H); \\ (x_1, x_2) \leftarrow \mathcal{A}_{\mathsf{cll\text{-}fnd}}(1^\lambda, r_H) \end{array} : \begin{array}{l} H(x_1) = H(x_2) \ \wedge \\ x_1 \neq x_2 \end{array} \right] \ .$$

Therefore we have reached a contradiction to collision resistance of $\mathcal{H}$, and it must be that $\mathbf{Exp}^{b,(0)}$ and $\mathbf{Exp}^{b,(1)}$ are computationally indistinguishable, i.e.,

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

A.2 Proof of Proposition 2

Let $b \in \{0,1\}$ be arbitrarily fixed. The only difference between games $\mathbf{Exp}^{b,(1)}$ and $\mathbf{Exp}^{b,(2)}$ is when $\mathcal{A}$ queries $\textsc{Eval}(\cdot)$ on x such that $H(x) = H(x^*)$. Then $\mathbf{Exp}^{b,(2)}$ aborts, while on any other query the oracle $\textsc{Eval}$ behaves equivalently in both games since $H(x) \neq H(x^*)$ implies $\mathsf{PF.Eval}(k_{h^*}, H(x)) = \mathsf{PF.Eval}(k, H(x))$.

We can therefore build an adversary $\mathcal{A}_{\mathsf{cll\text{-}fnd}}$ against the hash-function family $\mathcal{H}$ that on input $(1^\lambda, r_H)$ simulates $\mathbf{Exp}^{b,(2)}$ (except that instead of sampling H, it uses $H := \mathsf{H.Smp}(1^\lambda; r_H)$) until in an oracle query $\textsc{Eval}(x)$ the game would abort. $\mathcal{A}_{\mathsf{cll\text{-}fnd}}$ then outputs (x^*, x), which is a collision precisely when the game would have aborted.

A.3 Proof of Proposition 3

Assume towards contradiction that there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a polynomial $p(\cdot)$ such that for infinitely many λ

$$\left| \Pr[\mathbf{Exp}^{0,(2)}_{\mathcal{F},\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{1,(2)}_{\mathcal{F},\mathcal{A}}(\lambda) = 1] \right| \geq \frac{1}{p(\lambda)} \ .$$

Then we construct a PPT adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ running in game $\mathbf{Exp}^{\text{PCT-}b}_{\mathcal{PF},\mathcal{B}}(\lambda)$, the selective-security game of $\mathcal{PF}$ (cf. Fig. 2) as follows. (Note that $\mathcal{B}_2$ does not use its $\textsc{Eval}(\cdot)$ oracle.)

$\underline{\mathcal{B}_1(1^\lambda)}$

- $(x^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(1^\lambda)$.
- $H \leftarrow \mathsf{H.Smp}(1^\lambda)$ and $h^* := H(x^*)$.
- Return $(h^*, T := \{h^*\}, st := (H, x^*, st_{\mathcal{A}}))$.

$\underline{\mathcal{B}_2^{\textsc{Eval}(\cdot)}(st, k_{h^*}, y^*)}$ // y^* is either $\mathsf{PF.Eval}(k, H(x^*))$ or random.

- $crs \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
- $pp := (H, crs)$.
- $b' \leftarrow \mathcal{A}_2^{\textsc{Constr}(\cdot),\textsc{Eval}(\cdot)}(st_{\mathcal{A}}, y^*)$.
  - simulate $\textsc{Constr}(M)$:
    if $M \notin \mathcal{M}_\lambda \vee M(x^*) = 1$, reply $\bot$;
    else $P := P_{M,H,crs,k_{h^*}}$ as defined in (5); $\widetilde{P} \leftarrow \mathsf{diO}(1^\lambda, P_{M,H,crs,k_{h^*}})$.
    Return $k_M := (M, \widetilde{P}, (H, crs))$.
  - simulate $\textsc{Eval}(x)$:
    if $x = x^*$, reply $\bot$; if $H(x) = H(x^*)$ then abort;
    else reply $y := \mathsf{PF.Eval}(k_{h^*}, H(x))$.
- Output b'.

By construction $\Pr[\mathbf{Exp}^{\text{PCT-}b}_{\mathcal{PF},\mathcal{B}}(\lambda) = 1] = \Pr[\mathbf{Exp}^{b,(2)}_{\mathcal{F},\mathcal{A}}(\lambda) = 1]$, and therefore for infinitely many λ, it holds that

$$\left|\Pr[\mathbf{Exp}^{\text{PCT-0}}_{\mathcal{PF},\mathcal{B}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\text{PCT-1}}_{\mathcal{PF},\mathcal{B}}(\lambda) = 1]\right| \geq \frac{1}{p(\lambda)} \ .$$

This contradicts the selective security of PF, and we conclude that

$$\left|\Pr[\mathbf{Exp}^{0,(2)}_{\mathcal{F},\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{1,(2)}_{\mathcal{F},\mathcal{A}}(\lambda) = 1]\right| = \mathsf{negl}(\lambda) \ .$$